ডঃ দেবব্রত ঘোষ দস্তিদার
অরূপরতন ভট্টাচার্য

কমপিউটার পরিচিতি ও বেসিক ভাষা

ডঃ দেববত ঘোষ দস্তিদার
অধ্যাপক, কমপিউটার সায়েন্স
এ্যাণ্ড এঞ্জিনিয়ারিং বিভাগ
যাদবপুর বিশ্ববিদ্যালয়

ও

অরূপরতন ভট্টাচার্য

বেস্টবুক্‌স্‌
১এ কলেজ রো, কলিকাতা–৭০০ ০০৯

Computer Parichiti-O-Basic Vasha.
By Dr. Debabrata Ghosh Dastidar
and Arupratan Bhattacharya.

প্রথমপ্রকাশ ঃ জানুয়ারী ১৯৯১

প্রকাশক ঃ
বেস্ট বুক্স্
প্রকাশন বিভাগ
১এ কলেজ রো, কলিকাতা–৭০০ ০০৯

টাইপ কম্পোজিশন্ ঃ
টাইপোস্ ইণ্ডিয়া প্রাইভেট লিমিটেড
৬২/১ হিন্দুস্থান পার্ক, কলিকাতা–৭০০ ০২৯

ISBN : 81-85252-28-9.

মূল্য ঃ ৪৫ টাকা

ভূমিকা

আজ শিক্ষাক্ষেত্রে এবং কর্মজীবনে কমপিউটারের ব্যাপক অনুপ্রবেশ ঘটেছে। স্কুলের নীচু ক্লাসেই এর সূচনা এবং বৃহত্তর জীবনের সর্বত্রই এর প্রতিষ্ঠা। অথচ বিষয়টি সম্পর্কে অবহিত হওয়ার জন্যে যে প্রাথমিক জ্ঞানের প্রয়োজন, তা নিয়ে বাংলাভাষায় বই লেখার তেমন উদ্যোগ আজও পরিলক্ষিত হয়নি। বিক্ষিপ্তভাবে কিছু কিছু গ্রন্থ প্রকাশিত হয়েছে, ঠিকই। কিন্তু কমপিউটারের ভাষা-শিক্ষার সুপরিকল্পিত কোনো প্রচেষ্টা ইতিপূর্বে নজরে আসেনি। অবশ্য বিষয়টির অগ্রগতির সঙ্গে তাল রেখে ইংরেজি সহ বিদেশি বিভিন্ন ভাষায় বই লেখার কাজ চলেছে দ্রুতগতিতে।

কিন্তু বাঙ্গালি শিক্ষার্থীরা বাংলাভাষায় বিষয়টির সঙ্গে যথাযথ পরিচিত হবার সুযোগ পাবে না কেন? এমন একটি নিশ্চিত সম্ভাবনাপূর্ণ বিষয়-শিক্ষার ক্ষেত্রে ভাষা যাতে অন্তরায় হয়ে না দাঁড়ায় সেই প্রতিবন্ধকতা দূর করার উদ্দেশ্য নিয়েই বাংলাভাষায় এমন একটি বই প্রকাশের পরিকল্পনা গ্রহণ করেছি।

এই বইয়ে কমপিউটার সম্পর্কে বিশদভাবে আলোচনা করা হয়েছে। সূচনাপর্ব থেকে আজ পর্যন্ত কমপিউটার কোথায় এসে পৌঁছেছে? কিভাবে সে কাজ করে? কোনো সমস্যা সমাধানে তাকে নির্দেশ দেওয়ার পদ্ধতিই বা কি?

কমপিউটারের প্রাথমিক শিক্ষায় এই সব প্রশ্নের সহজ উত্তর জানা দরকার। আমাদের বইয়ে এই ধরনের প্রশ্নোত্তর নিয়ে যথাসম্ভব আলোচনা করা হয়েছে।

তা ছাড়া স্কুলে এবং শিক্ষা প্রতিষ্ঠানে বিভিন্ন ধরনের পার্সোনাল কমপিউটারে 'বেসিক' নামে যে উচ্চ পর্যায়ের ভাষা ব্যবহার করা হয়, সেই ভাষা সম্বন্ধেও এখানে বিস্তারিত আলোচনা আছে। আশা করি এই বই পড়ে ছোট ছোট সমস্যা সমাধানের জন্যে শিক্ষার্থীদের পক্ষে বেসিক ভাষায় প্রোগ্রাম লেখা সম্ভব হবে। সব শেষে বেসিকের

গ্রাফিক্স নিয়েও বইটিতে বলা হয়েছে। গ্রাফিক্সের নির্দেশাবলীর সাহায্যে প্রোগ্রাম লিখে কমপিউটারে তা চালিয়ে শিক্ষার্থীরা চিত্তাকর্ষক ছবিও আঁকতে পারবেন। মনে রাখা দরকার যে সব প্রোগ্রাম এখানে দেওয়া হয়েছে, তা পার্সোনাল কমপিউটারে চালানো সম্ভব। এ বই থেকে বেসিকের প্রায় সব নির্দেশাবলীরই প্রয়োগ সম্পর্কে একটা ধারণা করা যাবে। যাঁরা কমপিউটার বিষয়ে অল্পবিস্তর পরিচিত হতে চান, এই বইটি থেকে তাঁরা যথার্থ উপকৃত হবেন।

এই বইটি লেখার বিভিন্ন পর্যায়ে অনেকের সঙ্গে আলোচনা করেছি। এঁদের মধ্যে আছেন যাদবপুর বিশ্ববিদ্যালয়ের কমপিউটার সায়েন্স এবং ইঞ্জিনিয়ারিং বিভাগের অধ্যাপক মোহিত কুমার রায়, অধ্যাপক রাণা দত্তগুপ্ত এবং সেন্ট জেভিয়ার্স কলেজের ডঃ অশোক নাথ। এঁদের সকলের কাছে আমরা কৃতজ্ঞ।

দেবব্রত ঘোষ দস্তিদার
অরূপরতন ভট্টাচার্য

সূচীপত্র

কমপিউটার
ও
তার আনুষঙ্গিক যন্ত্রাংশ

ইতিহাস ঃ

কমপিউটার বেশি দিনের কথা নয়। তার আবির্ভাব আজ থেকে মাত্র বছর পঞ্চাশ আগে। কিন্তু হলে কি হবে, গণনার কাজে যন্ত্রের ব্যবহার বহুকাল ধরেই চলে আসছে। বিশ-পঁচিশ হাজার বছর পূর্বে মানুষের প্রথম সংখ্যা সম্বন্ধে ধারণা জন্মায় এবং সেই সময় থেকেই বিভিন্ন গণনা পদ্ধতিরও সূচনা লক্ষ্য করা যায়। প্রথমে মানব শরীরের সঙ্গে অঙ্গাঙ্গীভাবে যুক্ত আঙুলকে গণনার কাজে ব্যবহার করা হত। কিন্তু আঙুলের সংখ্যা নির্দিষ্ট। ফলে হিসেব এগোল না। এল পাথর ও নুড়ি। পাথর ও নুড়ি সহজে পাওয়া যায়, সংখ্যাতেও তা সীমাহীন। সুতরাং গণনার ক্ষেত্র বিস্তৃত হল। এর পরে এল যন্ত্র। তা এসে কুড়িয়ে পাওয়া পাথর ও নুড়ির স্থান অধিকার করলো। যন্ত্রের ব্যবহার সর্বপ্রথম লক্ষ্য করা যায় চীনদেশে আজ থেকে প্রায় পাঁচ হাজার বছর পূর্বে। এই যন্ত্রটির নাম 'অ্যাবাকাস' (Abacus)। একটি কাঠের বা ধাতুর কোনো ফ্রেমে কয়েক সারি তার – প্রত্যেক সারি তারের মধ্যে কয়েকটি পুঁতি থাকে। সংখ্যা সংক্রান্ত গণনা করা হয় এই পুঁতির সাহায্যেই। আজকের দিনেও কোনো কোনো দেশে এই যন্ত্রের ব্যবহার আছে। কিন্তু, সংখ্যা সংক্রান্ত গণনা নিয়ে চিন্তা-ভাবনা কখনো এক জায়গায় দাঁড়িয়ে থাকে নি। তা ধীরে ধীরে এগিয়ে গেছে। এবং গণনার ইতিহাসে শেষ পর্যন্ত দশমিক সংখ্যা পদ্ধতি-নির্ভর গণনার সূচনা হয়েছে। গণিতের ক্ষেত্রে এ এক যুগান্তকারী আবিষ্কার। এরপর 1700 খ্রীস্টাব্দের প্রথম দিকে জন নেপিয়ার 'স্লাইড রুল' উদ্ভাবন করেন। এটি জন নেপিয়ারের হাড় নামে পরিচিত। গণনার ক্ষেত্রে এই আবিষ্কারটিও উপেক্ষণীয় নয়।

এর সাহায্যে সহজেই দুটি বড় সংখ্যা গুণ করা সম্ভব। 'অ্যাবাকাস' এবং 'স্লাইড রুল' দু'টি যন্ত্রই হস্তচালিত।

অবশ্য এইসব হস্তচালিত যন্ত্রের সাহায্যে বড় বড় যোগ, বিয়োগ, গুণ ও ভাগফল নির্ণয় করা সহজ ছিল না। এই কারণে সময়ের ব্যবধানে এমন সব যন্ত্র তৈরির চেষ্টা হয় যার সাহায্যে বড় বড় গণনার কাজ সহজ এবং নিখুঁত ভাবে করা চলে। গণিতবিদ্ পাসকাল 1642 খ্রীস্টাব্দে একটি গণক-যন্ত্র আবিষ্কার করেন। এই যন্ত্রটির সাহায্যে খুবই দ্রুত 8 সারি সংখ্যা যোগ করা সম্ভব ছিল। গণিতবিদ্ লিব্‌নিজ 1673 খ্রীস্টাব্দে ওই যন্ত্রটির উন্নতি ঘটান এবং যোগের সঙ্গে বিয়োগ, গুণ ও ভাগেরও ব্যবস্থা করেন। তিনিই প্রথম গণক-যন্ত্রে 0 এবং 1 সংখ্যা দু'টির প্রয়োজনীয়তা বুঝতে পারেন।

গণিতের অগ্রগতির ক্ষেত্রে আর একটি উল্লেখযোগ্য ঘটনা স্বয়ংক্রিয় যন্ত্রের পরিকল্পনা। চার্ল্‌স ব্যাবেজ 1833 খ্রীস্টাব্দে প্রথম একটি স্বয়ংক্রিয় যন্ত্রের পরিকল্পনা করেন। এই যন্ত্রটির নাম দেওয়া হয় 'অ্যানালিটিকাল ইঞ্জিন' (Analytical Engine)। ব্যাবেজের পরিকল্পনা অনুসারে, ওই যন্ত্রটির সাহায্যে যোগ, বিয়োগ, গুণ ও ভাগ করা যাবে। তা ছাড়া বর্গমূল ও শতকরার হিসেব করাও সম্ভব হবে। কিন্তু এই যন্ত্রটি চার্ল্‌স ব্যাবেজ তাঁর জীবিতকালে সম্পূর্ণ করতে পারেন নি। তার প্রধান কারণ এই ধরনের যন্ত্রকে নিখুঁত করার জন্যে যে সব যন্ত্রাংশের প্রয়োজন, সেই সব যন্ত্রাংশ তৈরির খুব ভাল ব্যবস্থা তখনকার দিনে ছিল না। অবশ্য আধুনিক কালের কমপিউটারে চার্ল্‌স ব্যাবেজের পরিকল্পনাকে অনেকটাই কাজে লাগানো হয়েছে।

1890 খ্রীস্টাব্দে হার্মান হলারিথের তৈরি যন্ত্র আমেরিকার জনসংখ্যা গণনার কাজে ব্যবহার করা হয়। তিনি এক ধরনের কার্ড নিয়ে তার বিভিন্ন স্থানে ছিদ্রের সাহায্যে বিভিন্ন সংখ্যা বোঝাবার ব্যবস্থা করেন। ওই সছিদ্র কার্ডগুলিকে বলা হয় Punched Cards। হলারিথ একরকম যন্ত্রে এই কার্ড ব্যবহার করে গণনার কাজ করেছিলেন। 1890 খ্রীস্টাব্দের ডিসেম্বরে অস্ট্রিয়াতেও জনসংখ্যা গণনার কাজে এই ধরনের যন্ত্র এবং কার্ডের ব্যবহার লক্ষ্য করা যায়। এর পাঁচ বছর বাদে 1896 খ্রীস্টাব্দ থেকে ওই ধরনের পদ্ধতি অফিসের হিসেব-নিকেশের জন্য ব্যবহৃত হতে থাকে। হলারিথ 1896 খ্রীস্টাব্দে একটি কোম্পানি খোলেন। তিনি এর নাম দেন 'ট্যাবুলেটিং মেশিন কোম্পানি' (Tabulating Machine Com-pany)।

কিন্তু 1911 খ্রীস্টাব্দে এই কোম্পানিটির মালিকানার পরিবর্তন হয় এবং এরও কিছুকাল পরে এই কোম্পানিটি আরও দুটি কোম্পানির সঙ্গে মিলে একটি নতুন কোম্পানি শুরু করে। 192

খ্রীস্টাব্দে নতুন কোম্পানিটির নাম দেওয়া হল IBM (International Business Machines Corporation)। আজকের দিনেও কমপিউটার জগতে IBM একটি অতি পরিচিত নাম। কমপিউটার যুগের প্রথম দিকে এই সছিদ্র কার্ডের সাহায্যেই তথ্যাদি সরবরাহ করা হত। আমাদের দেশে কয়েক বছর পূর্বেও ওই কার্ডের প্রচুর ব্যবহার ছিল এবং এখনও হয়তো কোনো কোনা প্রতিষ্ঠানে এর ব্যবহার লক্ষ্য করা যাবে।

হলারিথের পদ্ধতি অনুসরণ করে 1944 খ্রীস্টাব্দে হারভার্ড বিশ্ববিদ্যালয়ে 'হারভার্ড-মার্ক'—নামে একটি সম্পূর্ণ স্বয়ংক্রিয় গণনার যন্ত্র তৈরি হয়। এটি তৈরি করতে পাঁচ বছরের বেশি সময় লাগে। আজকাল যে ধরনের কমপিউটার ব্যবহার দেখা যায়, তা সর্বপ্রথম তৈরি শুরু হয় 1942 খ্রীস্টাব্দে। দ্বিতীয় বিশ্বযুদ্ধ চলছে সে সময়ে। যুদ্ধের সঙ্গে গতির সম্পর্ক। খুব তাড়াতাড়ি অঙ্ক কষা সম্ভব তখন এমন একটি যন্ত্রের প্রয়োজন। এইজন্যেই 1942 খ্রীস্টাব্দে আমেরিকার সামরিক বিভাগের অন্তর্গত ক্ষেপনাস্ত্র গবেষণাগারের সহযোগিতায় পেনসিলভেনিয়া বিশ্ববিদ্যালয়ে একার্ট এবং মাউক্লির নেতৃত্বে কমপিউটার তৈরির কাজ শুরু হল। ফলে 1946 খ্রীস্টাব্দের 15 ফেব্রুয়ারি 'এনিয়াক' (ENIAC—Electronic Numerical Integrator And Calculator) জন্ম নিল। বলা যেতে পারে, আধুনিক কালের কমপিউটারের শুরু এই 'এনিয়াক' দিয়ে। কিন্তু কমপিউটারের বর্তমান অবস্থার জন্য বিখ্যাত গাণিতক ডঃ জন ভন নয়ম্যানের বিশেষ অবদান আছে। তিনিই প্রথম বলেন যে, কমপিউটারের সাহায্যে কোনো সমস্যার সমাধান করতে যে-সব নির্দেশের প্রয়োজন সে-সব নির্দেশ কমপিউটারের মধ্যে আগে থেকেই সঞ্চয় করে রাখা সম্ভব। কমপিউটারের অগ্রগতির ক্ষেত্রে এটি একটি অত্যন্ত উল্লেখযোগ্য পদক্ষেপ। নির্দেশগুলি সঞ্চয় করার পরে কমপিউটার স্বয়ংক্রিয়ভাবে এগুলি পালন করতে পারবে। ক্যালকুলেটারে যেমন একটি নির্দেশ পালন করার পর আর একটি নির্দেশ দিতে হয়, যোগের পরে গুণ বা গুণের পর বর্গমূল, কমপিউটারে তার প্রয়োজন নেই। সেখানে সব নির্দেশই দেওয়া হয় একসঙ্গে। ডঃ ভন নয়ম্যানের তত্ত্বের উপর ভিত্তি করে তৈরি হল সঞ্চিত কর্মসূচী (Stored Program) কমপিউটার 'এডভ্যাক' (EDVAC—Electronic Discrete Variable Automatic Computer)। এই যন্ত্রটির কাজ শেষ হওয়ার পূর্বেই কেমব্রিজ বিশ্ববিদ্যালয় আর একটি যন্ত্র উদ্ভাবন করে। এটির নাম 'এডস্যাক' (EDSAC—Electronic Delay Storage Automatic Calculator)।

কমপিউটারের বিভিন্ন পুরুষ ঃ

এরপর প্রযুক্তি বিজ্ঞানের প্রভূত উন্নতির ফলে কমপিউটার তৈরির ক্ষেত্রেও প্রচুর অগ্রগতি ঘটে। কমপিউটারে প্রথমদিকে ইলেকট্রনিক ভাল্‌ভ ব্যবহার করা হত। এই ধরনের কমপিউটারকে বলা হয় কমপিউটারের প্রথম পুরুষ (first generation)। 'এনিয়াক' বিশ্বের প্রথম ইলেকট্রনিক কমপিউটার। এই সব কমপিউটার আকারে বৃহৎ। 'এনিয়াক' কমপিউটারটি লম্বায় প্রায় 30 মিটার, প্রস্থে 1 মিটার এবং উচ্চতায় 3 মিটার। এটির বর্তনীতে (circuit) অন্যান্য ইলেকট্রিকাল যন্ত্রাংশ ছাড়া প্রায় 18000 ইলেকট্রনিক ভাল্‌ভ সন্নিবেশিত ছিল। প্রথম পুরুষ কমপিউটারের সময়কাল সাধারণত ধরা হয় 1940 থেকে 1952 পর্যন্ত। এরপর ট্রানজিস্‌টার আবিষ্কার–বিজ্ঞানের জগতে এ এক যুগান্তকারী পদক্ষেপ, এল ট্রানজিস্‌টারের যুগ। নাম ট্রানজিস্‌টার, কিন্তু আসলে এটি অর্ধ-পরিবাহী (Semi Conductor) 'ট্রায়োড' মাত্র।

ট্রানজিস্‌টার আবিষ্কার হওয়ার পরে কমপিউটারেও এর ব্যবহার ঘটতে আরম্ভ হয়। ট্রানজিস্‌টার ব্যবহৃত কমপিউটার দ্বিতীয় পুরুষ (Second generation) কমপিউটার। 1952 থেকে 1964-এর সময়কাল বলা চলে। ট্রানজিস্‌টার আকারে অনেক ছোট হওয়াতে দ্বিতীয় পুরুষ কমপিউটার প্রথম পুরুষ থেকে আকারে ছোট হয়ে এল অথচ তা রইলো দ্রুত কাজ করার ক্ষমতাসম্পন্ন। এখানে উল্লেখ করা যেতে পারে, আমাদের দেশে প্রথম ট্রানজিস্‌টার কমপিউটার তৈরি হয় 1966-তে। যাদবপুর বিশ্ববিদ্যালয় এবং ইণ্ডিয়ান স্ট্যাটিস্‌টিক্যাল ইন্‌সটিটিউটের মিলিত প্রচেষ্টায় এটি তৈরি হয়। এর নাম দেওয়া হয় 'ইসিজু্য–1' (ISIJU–1)।

ট্রানজিস্‌টারের পরে এল ইন্টিগ্রেটেড সার্কিট (সংক্ষেপে আই সি–IC)। এই সার্কিটকে একটি সূক্ষ্ম সার্কিট বলা যায়। খুব ছোট একটি অর্ধ-পরিবাহী পদার্থের মধ্যে 'ডায়োড', 'ট্রানজিস্‌টার'-এর মত অনেক বৈদ্যুতিক যন্ত্রাংশ জুড়ে যে সার্কিট বা বর্তনী তৈরি হয় তাকেই ইন্টিগ্রেটেড সার্কিট বলা হয়ে থাকে। আবার অর্ধ-পরিবাহী পদার্থের টুকরো 'সেমি কণ্ডাক্‌টর চিপ' নামে অভিহিত। প্রথম আই সি তৈরি হয়েছিল একটি সরু ও লম্বা জার্মেনিয়াম অর্ধ-পরিবাহী 'চিপ'-এর উপরে। একটি আই সি চিপ প্রস্থে 6 মিলিমিটার, লম্বায় 18 মিলিমিটার এবং উচ্চতায় 2 থেকে 3 মিলিমিটার। আই সি দিয়ে তৈরি কমপিউটার তৃতীয় পুরুষ (Third generation) কমপিউটার নামে পরিচিত। এরা আকারে অনেক ছোট অথচ অত্যন্ত দ্রুত কাজ করার ক্ষমতাসম্পন্ন। এর সময়কাল 1964 খ্রীস্টাব্দ থেকে 1972 খ্রীস্টাব্দ পর্যন্ত।

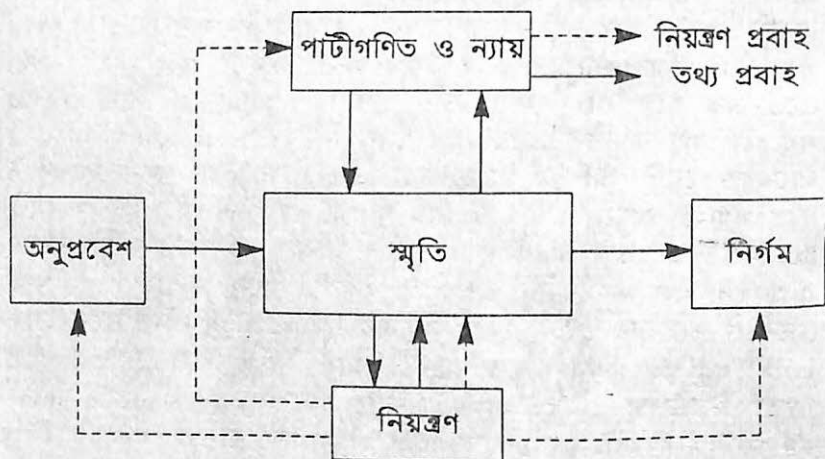
কমপিউটারের অগ্রগতি এখানেই স্তব্ধ হয়ে যায়নি। এরপরে আই সি তৈরির ক্ষেত্রে প্রভূত উন্নতি হতে থাকে। প্রথমদিকে খুব অল্প সংখ্যক যন্ত্রাংশের সমন্বয়ে এই আই সি তৈরি হত। একে 'স্মল স্কেল ইন্টিগ্রেশান' বা 'এস এস আই' বলা হয়। এখন 'লার্জ স্কেল ইন্টিগ্রেশান' (এল এস আই' ও 'ভেরি লার্জ স্কেল ইন্টিগ্রেশান' (ভি এল এস আই) করা সম্ভব। ভাবলে বিস্ময়ের কথা, আজকাল আকারে 5 বর্গ মিমি ও 1 মিমি বেধের আয়তনে 30,000-এর মত যন্ত্রংাশের সমন্বয় ঘটানো হয়। এর ফলে চতুর্থ পুরুষ (fourth generation) কমপিউটারের আবির্ভাব ঘটে। এই ধরনের কমপিউটার অনেক বেশি কাজ করার ক্ষমতাযুক্ত এবং আকারেও রীতিমতো ছোট। বর্তমানে পঞ্চম পুরুষ (fifth generation) কমপিউটারের কথা শোনা যায়। আশা করা যাচ্ছে, এই শতাব্দীর শেষ দশকে এই ধরনের কমপিউটারও বাজারে বেরোবে। কিন্তু বিজ্ঞানের উন্নতির সঙ্গে সঙ্গে কমপিউটার শেষ পর্যন্ত কত পুরুষে গিয়ে পৌঁছবে, তা লক্ষ্য করার বিষয়।

যত দিন যাচ্ছে, বিজ্ঞানের উন্নতিতে ইলেকট্রনিক্স যন্ত্র আকারে ক্রমশ খুব ছোট হয়ে আসছে। ফলে বর্তমানে কমপিউটার একটা পড়বার টেবিলের উপরেই বসানো যায়। তবে কমপিউটারের আকার ধরে তার শ্রেণীবিভাগ করলে বিভিন্ন কমপিউটারকে সাধারণভাবে চারভাগে ভাগ করা চলেঃ (1) খুব বড় (Super), (2) মাঝারি (midi), (3) ছোট (mini) এবং (4) মাইক্রো (micro)। একটি কমপিউটার এই চারভাগের কোনটিতে পড়বে তা নির্ভর করে সেই কমপিউটারের কাজ করার গতি, ক্ষমতা এবং কতটা তথ্য এতে এক সঙ্গে রাখা যায়, তার উপরে। আজকের দিনে বিশ্বের সবচেয়ে বড় কমপিউটারের নাম 'ক্রে' (cray)। এই কমপিউটারে 2 লাখেরও বেশি সার্কিট আছে এবং একটি সার্কিটে বেশ কয়েকটি করে যন্ত্রাংশ রয়েছে। এই ধরনের সুপার কমপিউটারে প্রতি সেকেণ্ডে 50 কোটিরও বেশি গণনা করা যায়। মাঝারি ধরনের কমপিউটারে প্রতি সেকেণ্ডে 4 কোটি থেকে 5 কোটি গণনা সম্ভব। মিনিতে সেকেণ্ডে 10 লাখ থেকে 1 কোটি এবং মাইক্রো-কমপিউটারে প্রায় 10 লাখের মত গণনা চলতে পারে। 'আই সি' পদ্ধতির উন্নতির ফলে 1971 খ্রীস্টাব্দে প্রথম 'মাইক্রো-প্রোসেসার' (micro-processor)-এর আবির্ভাব ঘটে। ফলে 'মাইক্রো' কমপিউটার ওই সময়ের কিছু পরেই বাজারে দেখা যায়। এই 'মাইক্রো-প্রোসেসার' একটি ছোট সিলিকন পদার্থের চিপে তৈরি। এটি আকারে 5 বর্গ মিমি এবং বেধে 1 মিমি। একটি চিপে কমপিউটারের কয়েকটি অংশ সমন্বিত। এইসব নতুন যন্ত্র বেরোনোর ফলে এখনকার একবারে ছোট কমপিউটারও ষাটের দশকের যে কোনো বড় কমপিউটারের

তুলনায় বেশি কার্যক্ষম। আমাদের দেশে এখন অনেক সংস্থাই 'মিনি এবং 'মাইক্রো' কমপিউটার তৈরি করেন।

কমপিউটারের বিভিন্ন অংশ ঃ

কমপিউটার সাধারনত দু ধরনেরঃ অনুরূপ (analog) এবং সংখ্যাত্মক (digital)। যে কমপিউটারে সংখ্যাকে অন্য কিছু দ্বারা প্রকাশ করা হয় তাকে অনুরূপ কমপিউটার বলে। যেমন, নেপিয়ারের 'স্লাইড রুল'-কে একটি অনুরূপ কমপিউটার বলা চলে। এখানে সংখ্যাকে দৈর্ঘ্য দ্বারা প্রকাশ করা হয়। আর যে সব কমপিউটারে সংখ্যা সংখ্যা হিসেবেই ব্যক্ত তাদের বলে সংখ্যাত্মক কমপিউটার। আজকাল কমপিউটারের মত ঘড়িও দু রকমের— অনুরূপ ও সংখ্যাত্মক। অনুরূপ ঘড়িতে আমরা ঘন্টা আর মিনিটের কাঁটার অবস্থান থেকে সময় ঠিক করি। সংখ্যাত্মক ঘড়িতে সময়টা সংখ্যা হিসেবেই দেখানো হয়ে থাকে। সেখানে ঘন্টা মিনিটের কাঁটা নেই। আমরা এখানে সংখ্যাত্মক কমপিউটার সম্বন্ধেই বিস্তারিত আলোচনা করাব।

একটি কমপিউটারে সাধারণত পাঁচটি মূল অংশ থাকে (1 সংখ্যক চিত্র)ঃ, স্মৃতি (memory), নিয়ন্ত্রণ (control), পাটীগণিত ও ন্যায় (arithmetic and logic), অনুপ্রবেশ (input) এবং নির্গম (output)।

১ সংখ্যক চিত্রঃ কমপিউটার ও তার পাঁচটি অংশ

কমপিউটারের সাহায্যে কোনো সমস্যা সমাধান করার প্রয়োজন হলে প্রথমে সমস্যাটি সমাধানের জন্য কমপিউটারের বোধগম্য কিছু

নির্দেশ (instruction) তৈরি করতে হয়। এই নির্দেশগুলি এবং সেই সঙ্গে যে সব তথ্য (data) এই ধরনের নির্দেশ কাজে লাগাবে প্রথমে তা অনুপ্রবেশ অংশের সাহায্যে স্মৃতিতে সঞ্চয় করা হয়। এই স্মৃতি কয়েক হাজার কোষ দিয়ে তৈরি। এই স্মৃতি কোষগুলি অনেকটা ব্যাঙ্কের সেফ ডিপজিট ভল্টের মত। ভল্টে যেমন বিষয়-সম্পত্তি (টাকা পয়সা, সোনার গহনা বা কোনো মূল্যবান কাগজপত্র) রাখা যায় তেমনি এখানে একটি কোষে একটি নির্দেশ কিংবা একটি তথ্য রাখা চলে। তবে কোনো কমপিউটারে একটি কোষে একটির বেশি নির্দেশ রাখা যায় আবার কোথাও একটি নির্দেশ রাখতে কয়েকটি কোষের দরকার। নির্দেশের সঙ্গে কোষের সম্পর্কটা কমপিউটারের গঠনের উপরে নির্ভর করে।

কিন্তু একটি কোষকে সনাক্ত করা হবে কি করে ? কমপিউটারে একটি কোষকে সংখ্যার দ্বারা সনাক্ত করা যায়। যে সংখ্যাটির দ্বারা কোষটিকে সনাক্ত করা হয় সেই সংখ্যাটিকে কোষটির ঠিকানা বলে। ভিন্ন ভিন্ন কমপিউটারে কোষের সংখ্যা বিভিন্ন এবং একটি কোষে কতটা তথ্য রাখা সম্ভব তাও এক একটি কমপিউটারের ক্ষেত্রে এক একরকম। যে কোনো কমপিউটারে কোনো নির্দেশ বা তথ্য 0 এবং 1-এর সাহায্যে রাখা হয়ে থাকে। এর কারণ কমপিউটারে সবকিছুই বৈদ্যুতিক যন্ত্রাংশের সাহায্যে করা হয়। এই ধরনের বৈদ্যুতিক যন্ত্রাংশের মধ্যে আছে ট্রানজিসটার, আই সি, অর্ধ-পরিবাহী পদার্থ, বৈদ্যুতিক তার। এদের প্রত্যেকের দ্বারাই কেবলমাত্র দুটি করে অবস্থান সূচনা করা সম্ভব। একটি ট্রানজিসটারের কথা ধরা যাক। ট্রানজিসটারটি হয় পরিবাহী হবে, না হলে অপরিবাহী; সে রকম একটি চৌম্বক পদার্থের চৌম্বকত্ব হয় একদিকে থাকবে, না হলে তার বিপরীত দিকে। একটি বৈদ্যুতিক তারের বেলায় কি হবে ? হয় তা বিদ্যুত-প্রবাহ যুক্ত হবে না হলে তাতে বিদ্যুৎ-প্রবাহ থাকবে না। কাজেই প্রত্যেকের ক্ষেত্রেই দুটি অবস্থার একটিকে 0 দিয়ে নির্দেশ করলে তার বিপরীত অবস্থাকে 1 দিয়ে বোঝানো চলে। এখানে বৈদ্যুতিক তারে বিদ্যুৎ-প্রবাহ না থাকলে তা 0 এবং থাকলে তা 1 দিয়ে বোঝানো হবে। একইভাবে বলা যায়, চৌম্বক পদার্থের কোনো একদিকের চৌম্বকত্বকে 0 দিয়ে নির্দেশ করলে বিপরীত দিকের চৌম্বকত্ব 1 দিয়ে নির্দেশ করা যাবে। কমপিউটারের ভাষায় 0 এবং 1 অঙ্ক দুটিকে বিট (bit) বলা হয়। Binary digit এই ইংরেজি শব্দ দু'টির প্রথমটির প্রথম অক্ষরটি এবং দ্বিতীয়টির শেষ দুটি অক্ষর নিয়ে এই bit শব্দটি গঠন করা হয়েছে। একটি কোষে বিটের সংখ্যা এক একটি কমপিউটারে এক এক রকম। সাধারণত একটি কোষে 8, 16, 24, 32, 48, কিংবা 64 টি বিট থাকে। তবে একটি কমপিউটারে

প্রত্যেকটি কোষে বিটের সংখ্যা নির্দিষ্ট। এই বিটের সংখ্যার উপর একটি কোষে তথ্যের পরিমাণ এবং সেই কমপিউটারের স্মৃতিতে স্মৃতিকোষের সংখ্যা নির্ভর করে।

একটি কমপিউটারে কত স্মৃতিকোষ আছে তা বোঝাবার জন্যে K অক্ষরটিকে ব্যবহার করা হয়। K সাধারণত 1024 (কোনো কোনো কমপিউটারে 1000)-এর সমান হিসেবে ধরা হয়ে থাকে। কাজেই একটি কমপিউটারে 32K স্মৃতিকোষ আছে বললে বোঝাবে তাতে স্মৃতিকোষের সংখ্যা 32×1024 বা 32768।

সাধারণত দু ধরনের স্মৃতি কমপিউটারে ব্যবহার করা হয়। এগুলি সিলিকন পদার্থের চিপ দিয়ে তৈরি। এক ধরনের স্মৃতিকে 'র‍্যাম' (RAM–Random Access Memory) বলে। এখানে কমপিউটার তথ্য সঞ্চয় করতে পারে, সেই সঞ্চিত তথ্য নিয়ে এসে কাজ করতে পারে এবং প্রয়োজনে এই তথ্য মুছে নতুন তথ্য সঞ্চয় করাও তার পক্ষে সম্ভব হতে পারে। কিন্তু এই ধরনের স্মৃতিযুক্ত কমপিউটারের অসুবিধে যখনই এর সুইচ বন্ধ করে দেওয়া হয় তখনই এর সঞ্চিত সমস্ত তথ্য মুছে যায়। একটি কমপিউটারের স্মৃতিতে কয়েক হাজার কোষ থাকা সম্ভব কিন্তু এর যে কোনো একটি কোষ থেকে তথ্য আনতে একই সময় লাগে। আবার তথ্য আনার মত যে কোনো একটি কোষে তথ্য রাখতেও সময়ের কোনো হেরফের ঘটে না। কোনো একটি কোষে তথ্য রাখবার সময়ে কোষটির ঠিকানার সাহায্যে তা সরাসরি সঞ্চয় করা সম্ভব।

দ্বিতীয় ধরনের স্মৃতি 'রম' (ROM–Read only Memory)। এই স্মৃতি তৈরির সময় যা সঞ্চয় করা হয় তা আর কোনো সময়ই সহজে মুছে যায় না, কমপিউটার বন্ধ করে দিলেও নয়। এর সঞ্চিত তথ্য বরাবরের মত থেকেই যায়। এই স্মৃতি থেকে সাধারণত যেমন কিছু মুছে ফেলা সম্ভব নয় তেমনি এই স্মৃতিতে কমপিউটার দিয়ে সহজে আর নতুন কিছু সঞ্চয় করাও চলে না। কমপিউটার যখন চালানো শুরু হয় তখন এই স্মৃতির সঞ্চিত নির্দেশের সাহায্যেই কমপিউটারটি কাজ আরম্ভ করতে পারে। কমপিউটারে নিয়ন্ত্রণ অংশটি সব কিছু নিয়ন্ত্রণ করে। কমপিউটারে সুইচ টিপলেই এই সব নির্দেশ 'রম' থেকে নিয়ন্ত্রণ অংশে এসে কাজ শুরু করে দেয়। কিন্তু নিয়ন্ত্রণ অংশে এই নির্দেশগুলি না এলে কমপিউটার অচল। তার পক্ষে তখন আর কিছু করা সম্ভব নয়। 'র‍্যাম'-এর সঙ্গে এর পার্থক্য এ ক্ষেত্রে সুইচ বন্ধ করলেও নির্দেশগুলি মুছে যায় না।

কমপিউটারে এই দুই ধরনের স্মৃতিই প্রয়োজন। যখন কমপিউটারের সাহায্যে কোনো সমস্যার সমাধান করা হয় তখন নির্দেশগুলি প্রথমে স্মৃতিতে সঞ্চয় করতে হয়। 'রম'-এ নতুন করে

আর কিছু রাখা যায় না বলে, যা কিছু রাখবার, তা কমপিউটার তৈরির সময়েই রাখতে হয়। কাজেই নতুন কোনো সমস্যা সমাধানের জন্য 'র‍্যাম' স্মৃতির দরকার। শুধু নতুন সমস্যার জন্য নয়, যে কোনো প্রক্রিয়ার ফল সাময়িকভাবে স্মৃতি কোষে সঞ্চয় করতে হলেও 'র‍্যাম' স্মৃতির প্রয়োজন। যে সব নির্দেশ বা তথ্য সব সমস্যার সমাধানের সময়েই প্রয়োজন, কমপিউটারে তা স্থায়ীভাবে রাখার ব্যবস্থা করতে হয় অর্থাৎ কমপিউটারের সুইচ বন্ধ করলেও এগুলি মুছে যাবে না। তাই এগুলি 'রম' স্মৃতির অন্তর্ভুক্ত। 'রম'-এ এই নির্দেশগুলি ছাড়াও আরও নানা ধরনের নির্দেশ ও তথ্য রাখার প্রয়োজন হয়ে থাকে। এ বিষয়ে পরে আলোচনা করা হবে।

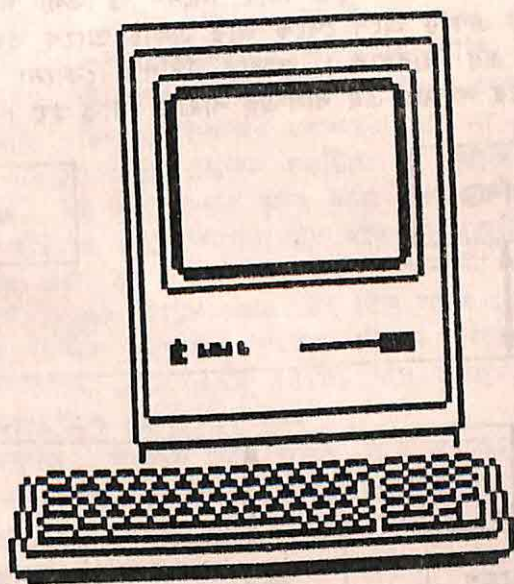
আজকাল সাধারণত 'ইপ্রম' (EPROM–Erasable Programable Read Only Memory) ধরনের 'রম' স্মৃতি ব্যবহার করা হয়। এক ধরনের যন্ত্র আছে। তার নাম 'প্রোগ্রামার'। এই স্মৃতি তৈরির সময়েই ওই যন্ত্রের সাহায্যে এতে কিছু নির্দেশ এবং তথ্য সঞ্চয় করা হয়। এই ধরনের স্মৃতিতে একবার যা সঞ্চয় করা হয় তা 'র‍্যাম' স্মৃতির মত খুব সহজেই মুছে যায় না। তবে স্মৃতির সঞ্চিত নির্দেশ এবং তথ্য মুছতে হ'লে একে আলট্রা-ভায়োলেট রশ্মির আওতায় সাধারণত 15 থেকে 20 মিনিট পর্যন্ত রাখা দরকার। তারপর তা মুছে ফেলে আবার 'প্রোগ্রামার' যন্ত্রটির সাহায্যে নতুন কোনো নির্দেশ এবং তথ্য সঞ্চয় করা সম্ভব। এখন অনেক সংস্থায় নানা ধরনের যন্ত্র কমপিউটারের সাহায্যে চালানো হয়। এই সব যন্ত্রে যে ধরনের অপারেশন করা চলে তার জন্য কমপিউটার তৈরির সময়েই 'ইপ্রম' স্মৃতিতে কিছু নির্দেশ সঞ্চয় করা হয়। এরপর কমপিউটারের সাহায্যে ওই যন্ত্রটিকে চালানোর সময়ে এর সব অপারেশন 'ইপ্রম' স্মৃতিতে সঞ্চিত নির্দেশগুলির সাহায্যেই করা সম্ভব। আবার যদি কোনো সময়ে ওই যন্ত্রটির কোনো অপারেশনের পরিবর্তনের প্রয়োজন হয় তবে প্রথমে পুরোনো নির্দেশগুলি আলট্রা-ভায়োলেট রশ্মির আওতায় এনে মুছে ফেলে এরপরে 'প্রোগ্রামার' যন্ত্রটির সাহায্যে আবার নতুন করে নির্দেশ এবং তথ্য সঞ্চয় করা হয়।

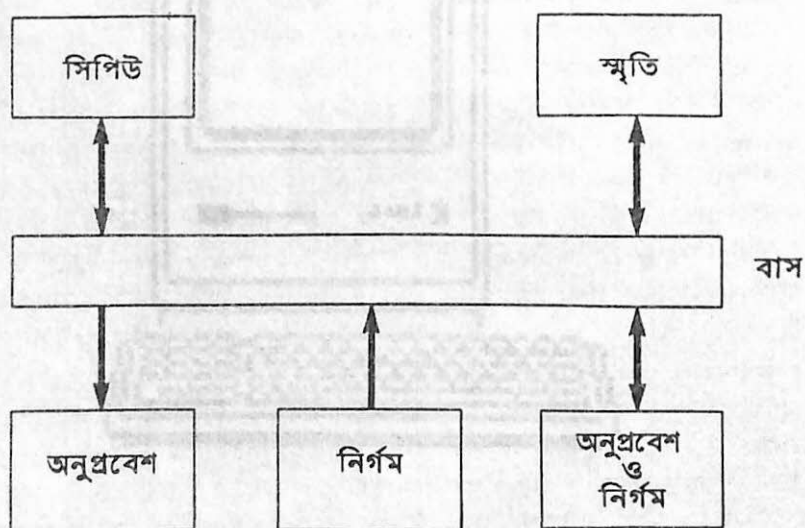
কমপিউটারে নিয়ন্ত্রণ এবং পাটীগণিত ও ন্যায় অংশ ইলেকট্রনিক সার্কিট দিয়ে তৈরি করা হয়। এখানে নির্দেশ এবং তথ্য কিছুক্ষণের জন্য রাখতে হলে খানিকটা জায়গার প্রয়োজন। এই জায়গা বিশেষ স্মৃতি অংশ হিসেবে চিহ্নিত, যদিও এদের মূল স্মৃতির অংশ হিসেবে ধরা হয় না। এই ধরনের স্মৃতিকে 'রেজিস্টার' (Register) বলে। এক একটি রেজিস্টার এক এক ধরনের কাজ করার জন্য ব্যবহার করা হয়। এই রেজিস্টারগুলি স্মৃতিকোষের ন্যায় কতকগুলি বিট দিয়ে তৈরি। একটি রেজিস্টারে কত বিট

আছে তা কমপিউটারের উপরে নির্ভর করে। ইলেকট্রনিক সার্কিট দিয়ে তৈরি বলে রেজিস্টারগুলির কাজ করার ক্ষমতা খুব দ্রুত। স্মৃতি থেকে তথ্য এখানে এনে তারপর তা দিয়ে কাজ করা হয়। নিয়ন্ত্রণ অংশ প্রথমে স্মৃতি থেকে একটি নির্দেশ এনে রেজিস্টারে রাখে। এরপর এই নির্দেশটি বিশ্লেষণ করে এটি কি ধরনের নির্দেশ তা স্থির করে। *নিয়ন্ত্রণ-অংশ যদি নির্দেশটি বিশ্লেষণ করে* পাটীগণিতের প্রয়োজন দেখে সেক্ষেত্রে নিয়ন্ত্রণ অংশ কাজটি করার জন্য পাটীগণিত ও ন্যায় অংশকে নির্দেশ পাঠায়। ধরা যাক, একটি নির্দেশে দুটি সংখ্যা যোগ করার কথা এবং যোগফলটি কোথায় রাখতে হবে, তা বলা আছে। এখানে দুটি সংখ্যা যোগ করা হচ্ছে বলে নির্দেশটিতে এটিও বলা থাকবে যে, সংখ্যা দুটি স্মৃতির কোন দুটি কোষে অবস্থিত। এই কোষ দুটির ঠিকানা অবশ্য পাটীগণিত ও ন্যায় অংশ নিয়ন্ত্রণ অংশের রেজিস্টার থেকে পেয়ে যাবে এবং সেই ঠিকানা অনুসারে সংখ্যা দুটি পাটীগণিত ও ন্যায়ের রেজিস্টারে এনে যোগ এই কাজটি সম্পন্ন করবে। নির্দেশটিতে একথাও বলা আছে যে, যোগ করার পর যোগফলটি স্মৃতির কোন কোষে রাখতে হবে। এই ঠিকানা ধরে নিয়ন্ত্রণ অংশ যোগফলটি পাটীগণিত ও ন্যায় অংশের রেজিস্টার থেকে নিয়ে এসে স্মৃতিতে রেখে দেবে। এইভাবে নিয়ন্ত্রণ অংশটি পরপর নির্দেশগুলি স্মৃতি থেকে নিয়ে আসে এবং বিশ্লেষণ অনুযায়ী কাজটি করার জন্য কোনো একটি অংশকে নির্দেশ পাঠায়। ফলাফল প্রকাশ করার সময় কিন্তু নিয়ন্ত্রণ অংশটি নির্গম অংশকে নির্দেশ দেয়। আবার কোনো তথ্য কমপিউটারের স্মৃতিতে আনতে হলে নিয়ন্ত্রণ অংশটিকে অনুপ্রবেশ অংশের সাহায্য নিতে হয়। কমপিউটারের সঙ্গে মানুষের যোগাযোগ ঘটে অনুপ্রবেশ এবং নির্গম অংশের মাধ্যমে। আজকাল অনেক কমপিউটারেই নিয়ন্ত্রন এবং পাটীগণিত ও ন্যায় অংশ দুটি একত্রে একটি অংশ হিসেবেই বর্তমান। এই দুটি একত্রে 'সিপিউ' (CPU–Central processing Unit) নামে অভিহিত। কমপিউটারে এক বা একাধিক বিভিন্ন ধরনের অনুপ্রবেশ অংশ এবং নির্গম অংশ থাকে। এমন অনেক যন্ত্র আছে যা অনুপ্রবেশ এবং নির্গম দুই ভাবেই ব্যবহার করা চলে। আসলে এই সব যন্ত্রে কিছু তথ্য সঞ্চয় করা যায় এবং এই সঞ্চিত তথ্য আবার প্রয়োজনে ব্যবহার করাও সম্ভব।

মাইক্রো-প্রোসেসারের কথা আগেই উল্লেখ করা হয়েছে। মাইক্রো-প্রোসেসার চিপে একটি 'সিপিউ'-এর পুরো অংশটি রাখা যায় এবং যে সব কমপিউটারের 'সিপিউ' একটি মাইক্রো-প্রোসেসর চিপ দিয়ে তৈরি তাদের মাইক্রো-কমপিউটার বলা হয়। এই কমপিউটারগুলি আয়তনে খুব ছোট এবং দামেও সস্তা। বর্তমানে মাইক্রো-প্রোসেসার প্রযুক্তির অগ্রগতির ফলে মাইক্রো-কমপিউটার

তৈরির ব্যাপারেও প্রভূত উন্নতি লক্ষ্য করা যায়। মাইক্রো-প্রোসেসার প্রথমে 8 বিট যুক্ত ছিল। অর্থাৎ এইসব সিপিউ-তে রেজিস্টারগুলি 8 বিটের। এরপর 16 বিট এবং এখন 32 বিট-এর মাইক্রো-প্রোসেসারও পাওয়া যায়। ফলে আজকাল মাইক্রো-কমপিউটারগুলি অনেক বড় কমপিউটারের সমান কাজ করতে পারে, যদিও এদের দাম তুলনায় অনেক কম।

বর্তমানে IBM সংস্থা 16 বিটের মাইক্রো-প্রোসেসার দিয়ে যে সব কমপিউটার তৈরি করে, বাজারে তারা পার্সোনাল কমপিউটার (PC) নামে অধিক পরিচিত এবং এরা খুব জনপ্রিয়। এই কমপিউটারে 'রম' এবং 'র‍্যাম'—দু'রকমেরই স্মৃতি থাকে। 'রম' সাধারণত 8K কোষযুক্ত। আজকাল অন্যান্য অনেক সংস্থাই এই ধরনের PC তৈরি করে। কিন্তু IBM PC অত্যধিক জনপ্রিয় বলে যাতে IBM PC-র সঙ্গে এই সব সংস্থার তৈরি কমপিউটারের মিল থাকে, সেদিকেও লক্ষ্য রাখা হয়। এই পার্সোনাল কমপিউটার আবার কয়েক ধরনের, PC, PC/XT এবং PC/AT। PC-র চেয়ে PC/XT বেশি শক্তিশালী এবং PC/AT আবার PC/XT থেকে অধিক শক্তিসম্পন্ন।

২ সংখ্যক চিত্র ঃ একটি পার্সোনাল কমপিউটার

কমপিউটারের প্রত্যেকটি আলাদা অংশ একে অন্যের সঙ্গে তারের সাহায্যে যোগাযোগ রক্ষা করে। তথ্যকে এক অংশ থেকে অন্য অংশে পাঠাতে এই যোগাযোগের প্রয়োজন। প্রথমেই অনুপ্রবেশ অংশের সাহায্যে নির্দেশ স্মৃতিতে নিয়ে আসা হয়। এই অনুপ্রবেশ অংশটির কাজের সূচনাও নিয়ন্ত্রণ অংশটি করায়। এরপর নিয়ন্ত্রণ অংশটি স্মৃতি থেকে নির্দেশ নিয়ে এসে সেই নির্দেশ অনুযায়ী অনুপ্রবেশ, পাটীগণিত ও ন্যায় অথবা নির্গম অংশকে নির্দেশ পাঠায়। আগেকার দিনে একটি কমপিউটারে একটি অংশর সঙ্গে আর একটি অংশর যোগাযোগ যেভাবে থাকতো 1 সংখ্যক চিত্রে তা দেখানো হয়েছে। এখানে দুরকমের রেখা দেখতে পাওয়া যাচ্ছে। এক ধরনের রেখা ব্যবহার করা হয়েছে যার মাধ্যমে একটি অংশ থেকে অপর এক বা যে কোনো অংশে তথ্য পাঠানো হয়। অন্য আর এক ধরনের রেখার সাহায্যে কেবলমাত্র নিয়ন্ত্রণ অংশ থেকে অন্যান্য অংশে তথ্য পাঠানোর কথা বোঝানো হয়েছে। আজকাল অবশ্য এ রকম যোগাযোগ নজরে আসে না। এখন সাধারণত যে ধরনের যোগাযোগ লক্ষ্য করা যায় তা 3 সংখ্যক চিত্র দেখলে বোঝা যায়। এই চিত্রে 'বাস' অনেকটা রাজপথের মত যার সঙ্গে সব কটি অংশেরই যোগাযোগ আছে। এই 'বাস' অনেক সারি সারি তার দিয়ে তৈরি যার মধ্যে দিয়ে নির্দেশ ও তথ্য পাঠানো যায়। যে কোনো একটি অংশ থেকে আর একটি অংশে তথ্য আদান-প্রদানের জন্যে এর ব্যবহার। আবার সিপিউ কোনো সংকেত কোথাও পাঠাবার সময়েও এই বাস-এর সাহায্য নিতে হয়।

৩ সংখ্যক চিত্র ঃ একটি আধুনিক কমপিউটার

কমপিউটারের আনুষঙ্গিক যন্ত্রাংশ ঃ

কমপিউটারে নানা ধরনের যন্ত্র অনুপ্রবেশ এবং নির্গম অংশ হিসেবে ব্যবহার করা হয়। এবারে সেরকম কিছু যন্ত্র সম্বন্ধে আলোচনা করা যাক।

1. কার্ড রীডার (Card reader)

কমপিউটারের প্রথম যুগে কোনো সমস্যার সমাধানের জন্যে যে সব নির্দেশের প্রয়োজন হত এবং নির্দেশগুলি যে সব তথ্যের সাহায্যে কাজ করতো স্মৃতিতে তা সঞ্চয় করার জন্য হলারিথের কার্ডের চল ছিল। এই কার্ড সাধারণত দৈর্ঘ্যে 19.3 সেমি, প্রস্থে 9.5 সেমি এবং বেধে 0.018 সেমি। একটি কার্ডে পাশাপাশি পরপর 80টি স্তম্ভ (Coloumn) এবং আড়আড়ি বা লম্বালম্বি 12টি সারি (Row) থাকে। এই 12টি সারির মধ্যে 9 সংখ্যক সারিটির স্থান সকলের নীচে। এর উপরে 8 সংখ্যক, তার উপরে 7 এবং এইভাবে 1 সংখ্যক সারিটি অবস্থিত। কার্ডে যে 80টি স্তম্ভ আছে পাশাপাশি, তার প্রত্যেকটিতে উপর থেকে নীচে প্রথম সারি থেকে নবম সারি পর্যন্ত 1 থেকে 9 ছাপানো থাকে। তার অর্থ প্রথম সারিতে আছে 80টি 1, দ্বিতীয় সারিতে 80টি 2 এবং ক্রমে নবম সারিতে 80টি 9। এইভাবে প্রতিটি কার্ডেই 80টি 1, 80টি 2, এবং শেষ পর্যন্ত একইভাবে 80টি 9 ছাপানো আছে। কার্ডে পাশাপাশি 80টি স্তম্ভ থাকাতে একটা কার্ডে মোট 80টি অক্ষর বোঝানো যেতে পারে। কিন্তু একটি স্তম্ভে 1টির বেশি অক্ষর বোঝানো যায় না এবং 1 টি অক্ষরের জন্য 12 টি সারির কোনো একটিতে বা একাধিক সারিতে ছিদ্র করা হয়। এই ছিদ্র করার জন্য নানা প্রকারের যন্ত্র পাওয়া যায়। কোন অক্ষরের জন্য কোথায় ছিদ্র হবে তা হলারিথের কার্ড কোড অনুসারে করা হয়। এই কার্ড কোড অনেকটা টেলিগ্রাফের মর্স কোডের মত। মর্স কোডে যেমন ভিন্ন ভিন্ন অক্ষর বোঝাতে ডট্ এবং ড্যাশের বিভিন্ন সমন্বয়ের সাহায্য নেওয়া হয়েছে, কার্ডেও তেমনি 12টি সারির কোন কোন সারিতে ছিদ্র আছে তার উপরে নির্ভর করে অক্ষরটিকে বোঝানো হয়।

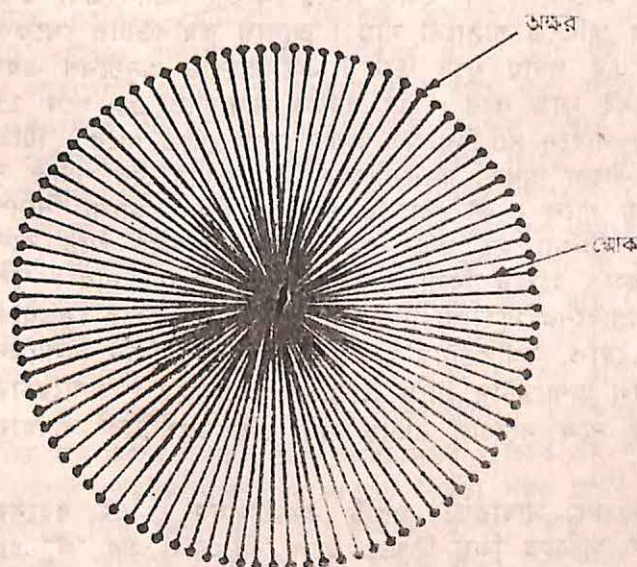
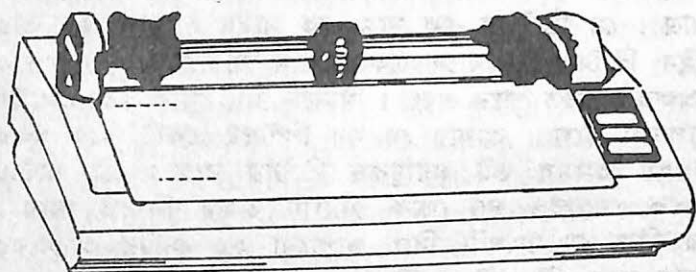
কার্ডে যেখানে 1 সংখ্যক সারি আছে, তার উপরের সারিতে 0 ছাপানো থাকে। এটিকে সাধারণত 10 সংখ্যক সারি বলা হয়। তার উপরে বেশ খানিকটা জায়গা আছে। হিসেব করলে সেখানে তিনটি সারির মত জায়গা পাওয়া যাবে। সেখানে কিছু ছাপানো থাকে না বলে অঞ্চলটিকে ফাঁকা বলা যায়। 10 সংখ্যকের উপরের সারিটিকে 11 সংখ্যক এবং তার উপরটিকে 12 সংখ্যক বলে চিহ্নিত করা হয়। 12 সংখ্যক সারির উপরে তাহলে একটি সারির মত জায়গা ফাঁকা পাওয়া যায়। এই জায়গাটি রাখার কারণ, কোনো কোনো কার্ড

ছিদ্র করার যন্ত্রে অক্ষর ছাপানোর ব্যবস্থা থাকে। কাজেই যে অক্ষরটির জন্য ছিদ্র করা হল সেই অক্ষরটিকে ওই জায়গাতে ছাপানো হয়। এর ফলে ঠিক অক্ষরটির জন্য ছিদ্র করা হয়েছে কিনা পরে তা মিলিয়ে দেখা যেতে পারে। আমরা যদি ইংরেজি বর্ণমালার A অক্ষরটির কথা বলি, তাহলে হলারিথের কোড অনুসারে কোথায় ছিদ্র থাকবে ? হলারিথের কোডে A বোঝাতে 12 সংখ্যক সারিতে এবং 1 সংখ্যক সারিতে ছিদ্র করতে হবে। অনুরূপভাবে B বোঝাতে 12 সংখ্যক সারিতে এবং 2 সংখ্যক সারিতে ছিদ্র করা হয়। কিন্তু A, B, C, D-এর মত বর্ণ ছাড়া কোনো সংখ্যা বোঝানোর সময়ে কি হবে ? দশমিক সংখ্যা 5-এর কথা ধরলে তার জন্যে কেবলমাত্র 5 সংখ্যক সারিতেই ছিদ্র এবং এইভাবে 9 বোঝানোর জন্য সর্বশেষ 9 সংখ্যক সারিতেই ছিদ্র করা হয়ে থাকে। কাজেই একটি সারিতে কোন অক্ষরটি বোঝানো হয়েছে তা ছিদ্রগুলির অবস্থান থেকেই জানা যায়। এখানে একটা কথা মনে রাখতে হবে যে, কার্ডের উপরে ছিদ্র অনুযায়ী অক্ষরটি ছাপানো থাকলে কি হবে কমপিউটার তা দেখতে পায় না, কেবলমাত্র ছিদ্রগুলির মধ্য দিয়ে বিদ্যুত-প্রবাহের ফলেই অক্ষরটিকে সনাক্ত করতে পারে।

একটি নির্দেশে যে সব অক্ষর থাকে হলারিথের সংকেত অনুযায়ী কার্ডে ছিদ্রের সাহায্যে তা নির্দিষ্ট করা হয়। অনেক সময়েই একটি সমস্যা সমাধানের জন্য যে সমস্ত নির্দেশের প্রয়োজন তাতে কয়েক শত কার্ডের দরকার হতে পারে। এই সব কার্ডে ছিদ্র করার পর 'কার্ড রীডারে' তা জমা পড়ে। 'কার্ড রীডার'-এর মাধ্যমে এক একটি কার্ড পড়া হয় এবং মুদ্রিত নির্দেশ কমপিউটারের স্মৃতিতে সঞ্চয় করা হয়। 'কার্ড রীডার'-এর ব্যবহার অনুপ্রবেশ অংশ হিসেবে। একটি কার্ডে একবার একটি নির্দেশের জন্য ছিদ্র করলে আর অন্য কোনো নির্দেশের জন্য তা ব্যবহার করা চলে না। ফলে আজকাল এর প্রচলন খুবই কম। আমাদের এখানে এই ধরনের ব্যবহৃত কার্ড পরে বেসরকারি বাসের টিকিট হিসেবে দেখতে পাওয়া যায়। একটি কার্ড থেকে প্রায় 20/25টি টিকিট করা চলে।

2. প্রিন্টার (Printer)

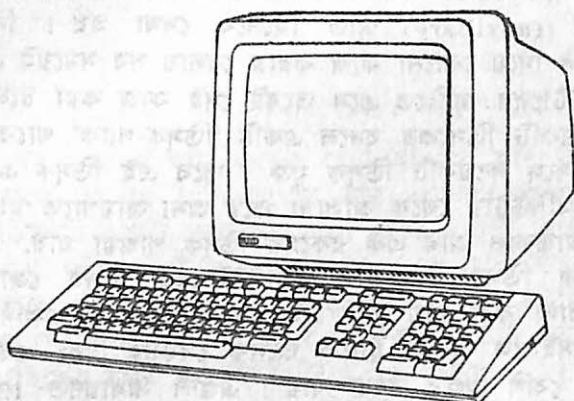
প্রিন্টারে যে কাগজ লাগানো দেখা যায় তাতে কমপিউটারের সাহায্যে সমাধান করা বিভিন্ন সমস্যার ফল ছাপানো হয়ে থাকে। এই প্রিন্টার সাধারণত দু ধরনের। এক ধরনের প্রিন্টারে অক্ষর এবং কাগজের মধ্যেকার মসীলিপ্ত ফিতের উপর অক্ষরগুলির আঘাতের ফলেই ছাপ ফুটে ওঠে এবং অন্য আর এক ধরনে বিনা আঘাতেই এই অক্ষরের ছাপ ফোটানো হয়। দ্বিতীয় ধরনের প্রিন্টারে মেকানিক্যাল যন্ত্রাংশ না থাকায় এরা খুব দ্রুত কাজ করতে পারে কিন্তু এদের দাম

খুব বেশি। এই রকমের প্রিন্টারের উদাহরণ হিসেবে 'লেসার' প্রিন্টারের কথা উল্লেখ করা যেতে পারে। এখানে ছাপানোর কাজ চলে 'লেসার' প্রয়োগ করে। প্রথম ধরনের প্রিন্টারকে আবার দু ভাবে ভাগ করা সম্ভব। এর একটিতে এক লাইনের সব কটি অক্ষর একসঙ্গে ছাপানো যায় এবং অন্যটিতে ছাপানো সম্ভব একটি করে অক্ষর মাত্র। যে প্রিন্টারে এক সঙ্গে এক লাইন ছাপানো হয় তার নাম লাইন প্রিন্টার। এখানে এক লাইনে সাধারণত 120 থেকে 132টি অক্ষর থাকতে পারে এবং 1 মিনিটে 300 থেকে 1400 লাইন পর্যন্ত ছাপানো চলে। আবার যে সব প্রিন্টারে একটি করে অক্ষর ছাপানো হয় তাদের ডট ম্যাট্রিক্স প্রিন্টার বলে। এই জাতীয় প্রিন্টার এক সেকেণ্ডে 80 থেকে 200টি অক্ষর লিখতে পারে। প্রিন্টারের ঠিক যে অংশটি দিয়ে ছাপানো হয় সেটির আকারের উপর নির্ভর করে এই 'ডট্ ম্যাট্রিক্স' প্রিন্টারের নানা রকমের নাম দেওয়া হয়, যেমন, ডেইজী হুইল, গল্‌ফ বল। এই নাম দেখে বুঝতে অসুবিধে হয় না প্রিন্টারের এই অংশটি দেখতে কি রকম। এখানে একটা কথা বলা দরকার, 'কার্ড রীডার'কে যেমন অনুপ্রবেশ অংশে হিসেবে ধরা হয়, প্রিন্টার তেমনি নির্গম অংশ হিসেবে ব্যবহৃত। আজকাল যে সব পার্সোনাল কমপিউটার বাজারে বেরিয়েছে, তাতে ডট ম্যাট্রিক্স প্রিন্টারই লাগানো হয়। মিনি এবং তার চেয়ে বড় ধরনের কমপিউটারের সঙ্গে লাইন প্রিন্টার থাকে।

৪ সংখ্যক চিত্র ঃ ডেইজী হুইল

৫ সংখ্যক চিত্র ঃ একটি ডট্-ম্যাট্রিক্স প্রিন্টার

3. কনসোল (Console)

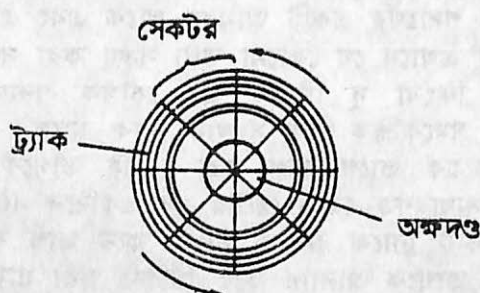
কনসোলে সাধারণত একটি 'কী বোর্ড' এবং একটি টিভির মত পর্দা থাকে। কনসোলের 'কী বোর্ড' দিয়ে কোনো নির্দেশ এবং তথ্য কমপিউটারের স্মৃতিতে পাঠানো যায়। আবার কমপিউটার থেকেও কিছু নির্দেশ এর পর্দায় ফুটে উঠে। এই অংশটি অনুপ্রবেশ এবং নির্গম – দুভাবেই কাজ করে। এর পর্দাতে সাধারণত এক সঙ্গে 25 লাইন এবং 1 লাইনে 80 টির মত অক্ষর দেখা যেতে পারে। টিভি যেমন সাদা-কালো অথবা রঙ-বেরঙের হয় তেমনি এর পর্দাও দু রকমেরই হয়ে থাকে। কী বোর্ডের সাহায্যে যখন কোনো নির্দেশ কমপিউটারে পাঠানো হয় তখন তা এর পর্দায় ফুটে ওঠে বলে নির্দেশটি ঠিকমত হয়েছে কিনা সহজেই দেখে নেওয়া যায়। ভুল হলে ভ্রম সংশোধনের সাহায্যে নির্দেশটি আবার ঠিক করে দেওয়াও সম্ভব। যাই হোক, কমপিউটার চালক-নির্ভর যন্ত্র। যিনি চালাবেন তাঁকে কনসোল অপারেটার নামে অভিহিত করা হয়। অপারেটার কমপিউটারের সঙ্গে সংযোগ রক্ষার জন্য এই অংশটিকে ব্যবহার করে থাকেন।

প্রত্যেকটি PC সাধারণত একটি কনসোলযুক্ত। এই ধরনের কমপিউটারের স্মৃতিতে কিছু লিখতে হলে 'কী-বোর্ড'-এর 'কী'-এর দ্বারাই তা করা সম্ভব। 'কী' সাধারণত দু রকমের – অক্ষরের 'কী' এবং নিয়ন্ত্রণের 'কী'। ইংরেজি বর্ণমালার ছোট-বড়-অক্ষর, দশমিক

সংখ্যা পদ্ধতির দশটি অঙ্ক এবং বিশেষ ধরনের চিহ্নকে অক্ষরের 'কী' বোঝানো হয়। কিন্তু 'কী-বোর্ড'-এর বিভিন্ন নিয়ন্ত্রণের জন্যে অন্য 'কী' অর্থাৎ 'নিয়ন্ত্রণ কী'-এর প্রয়োজন। এই ধরনের 'কী-বোর্ড'-এর বিস্তারিত আলোচনা পরে করা হবে।

৬ সংখ্যক চিত্র : কনসোল

4. ম্যাগনেটিক ডিস্ক (Magnetic Disk)

এটিও কনসোলের মত অনুপ্রবেশ এবং নির্গম অংশ – দু ভাবেই ব্যবহার করা হয়। এই ডিস্ক একটি গোলাকার অ্যালুমিনিয়ামের প্লেট দিয়ে তৈরি। প্লেটটিতে 'ফেরিক অক্সাইড' বা 'ক্রমিয়াম অক্সাইড' চৌম্বক পদার্থের একটি আবরণ থাকে এবং এই চৌম্বক পদার্থের সাহায্যেই এখানে যে কোনো তথ্য সঞ্চয় করা সম্ভব। এই প্লেটের একদিকে কিংবা দু দিকেই এই চৌম্বক পদার্থ থাকতে পারে। প্লেটটিতে সমকেন্দ্রিক কিছু সংখ্যক ট্র্যাক আছে। প্রত্যেকটি ট্র্যাকই আবার কয়েক ভাগে ভাগ করা। এই ভাগকে 'সেকটর' (Sector) বলে। সাধারণত একটি প্লেটের এক একদিকে 400 টি করে ট্র্যাক এবং প্রত্যেকটি ট্র্যাকে আবার 60 টি করে ভাগ বা সেকটর থাকে। প্রত্যেকটি ভাগকে আলাদা করে চিহ্নিত করা যায়। একটি ভাগে 200 টির মত অক্ষর সঞ্চয় করা চলে। তবে আজকাল অনেক ডিস্ক আছে যেখানে একটি ভাগে অনেক বেশি অক্ষরও রাখা যেতে পারে। এই রকমের ডিস্ক থেকে কমপিউটারে সেকেণ্ডে 500 হাজারটির মত অক্ষর আদান-প্রদান করা সম্ভব। তবে এই ডিস্কের কোনো একটি সেকটর খুঁজে বের করতে কিছুটা সময় লেগে যায়। যখন এখানে কিছু সঞ্চয় করা হয় কিংবা এখান থেকে কিছু তথ্য কমপিউটারের স্মৃতিতে আনা হয় এই ডিস্ক তখন চক্রাকারে ঘুরতে থাকে। কাজেই সঠিক সেকটরটি বের করতে যে সময় লাগে সেটি সাধারণত 30 মিলি সেকেণ্ডের মত। এই ডিস্কের যে কোনো

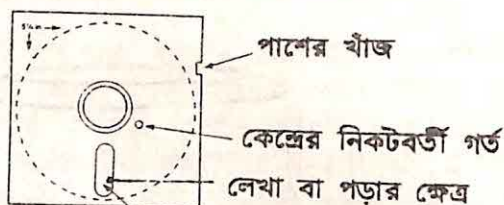
সেকটর থেকে প্রয়োজন মত তথ্য সরাসরি কমপিউটারে আনা যায়। ডিস্‌কের ডাইরেকটরি থেকে তার ঠিকানা জেনে নিয়ে এখানে সরাসরি চলে আসা সম্ভব। এগুলি দেখতে অনেকটা গানের রেকর্ডের মত। একটি ডিস্‌কে অনেক তথ্য ধরানো চলে এবং এই অংশটিকে একটি সহযোগী (auxiliary) স্মৃতি হিসেবে দেখা হয়। কিন্তু ডিস্‌কের তথ্যগুলি দিয়ে কোনো কাজ করার বেলায় সব সময়েই এই তথ্যগুলি কমপিউটারের স্মৃতিতে এনে তবেই সেই কাজ করা চলে। অনেক সময়েই একটি ডিসকের বদলে একটি 'ডিস্‌ক-প্যাক' থাকে। ডিস্‌ক-প্যাক একসঙ্গে কয়েকটি ডিস্‌ক যুক্ত। তবে এই ডিস্‌ক এবং ডিস্‌ক-প্যাক কমপিউটার থেকে আলাদা করে অন্য জায়গাতে রাখা যেতে পারে। আজকাল আর এক রকমের ডিস্‌ক পাওয়া যায়, এর নাম 'উইনচেষ্টার ডিস্‌ক'। এগুলি কমপিউটারের সঙ্গেই জোড়া এবং এদের আলাদা করা যায় না। এই ধরনের ডিস্‌কের একদিকে অনেক ট্র্যাক, সেইসঙ্গে একটি ট্র্যাকে অনেক সেকটর এবং একটি সেকটরে অনেক বেশি অক্ষর রাখা যায়। এগুলি সাধারণত 13.3 সেমি, 20.3 সেমি কিংবা 35.6 সেমি ব্যাসযুক্ত। এখানে 2 কোটি থেকে 100 কোটি অক্ষর রাখার জায়গা আছে। বর্তমানে ডিস্‌ক তৈরির ব্যাপারে এত উন্নতি লক্ষ্য করা যাচ্ছে যে, কিছুদিনের মধ্যে আরও অনেক বেশি তথ্য এতে রাখা সম্ভব হবে, আশা করা যায়। এই ডিস্‌ক দিয়ে কিছু করতে হলে একে একটি যন্ত্রের সাহায্য নিতে হয়। এই যন্ত্রটিকে ডিস্‌ক ড্রাইভ বলা হয়।

৭ সংখ্যক চিত্র : ডিস্‌কের সেকটর ও ট্র্যাক

আর এক ধরনের ডিস্‌কের কথাও বলা যায়, যেগুলি বর্তমানে ছোট কমপিউটারে বা পার্সোনাল কমপিউটারে ব্যবহার করা হচ্ছে। এদের ফ্লপি ডিস্‌ক বলে। এরা আকারে অনেক ছোট। আজকাল 13.3 সেমি ফ্লপি ডিস্‌কেরই বেশি প্রচলন। এর দু দিকেই তথ্য রাখা যায়। একটি ফ্লপিতে সাড়ে তিন লাখ মত অক্ষর রাখা

৮ সংখ্যক চিত্র ঃ ডিস্‌ক-প্যাক

সম্ভব। শোনা যায়, আরও ছোট আকারে ফ্লপি উদ্ভাবিত হবে অথচ তাতে তুলনায় বেশি তথ্যের স্থান সংকুলান সম্ভব হবে। এই ফ্লপি ডিস্‌কটি একটি কভারের মধ্যে থাকে। এটি 'মাইলার' দিয়ে তৈরি এবং পৃষ্ঠদেশ চৌম্বক পদার্থ যুক্ত। এই ফ্লপি ডিস্‌ক ব্যবহার করতে হলে এটিকে একটি ডিস্‌ক-ড্রাইভের উপর এমনভাবে রাখতে হয় যাতে ফ্লপির কেন্দ্রস্থলে যে বৃত্তাকার গর্তটি আছে সেটি ডিস্‌ক-ড্রাইভের শঙ্কুর মত জিনিসটিতে যেন ঠিকমত বসে যায়। এই ডিস্‌ক ড্রাইভের সঙ্গে সঙ্গে ফ্লপিও মিনিটে 300 বার ঘোরে। ফ্লপি ঘোরার সময়ে এ থেকে কিছু পড়া বা এতে কিছু লেখার যন্ত্রটি এগিয়ে পিছিয়ে গিয়ে ঠিক জায়গামত পৌঁছে যায় এবং প্রয়োজন মত কিছু পড়ে বা লেখে। ফ্লপি ডিস্‌ক ব্যবহার করার আগে এতে কয়েকটি নির্দেশ দিয়ে কমপিউটারের সাহায্যে এর পৃষ্ঠদেশে কতগুলি বৃত্তাকার ট্র্যাক এবং প্রত্যেকটি ট্র্যাকে আবার কিছু সংখ্যক সেকটর তৈরি করা হয়। প্রত্যেকটি ডিস্‌কেই একটি নির্ধারিত ট্র্যাক আছে যেখানে ডিস্‌কের কোন সেকটর কি তথ্যযুক্ত সেই সংবাদটি ধরা থাকে। কাজেই প্রয়োজনে এই ট্র্যাকের সাহায্যে কোনো একটি ট্র্যাকের বিশেষ কোনো সেকটরে সরাসরি যাওয়া সম্ভব। এই ডিস্‌কের কেন্দ্রস্থলের কাছে একটি গর্ত অবস্থিত। এটির সাহায্যে কমপিউটার প্রথম সেকটরটি খুঁজে বের করতে পারে এবং এর সাহায্যেই সরাসরি অন্যান্য সেকটরের তার পক্ষে চলে যাওয়া সম্ভব।

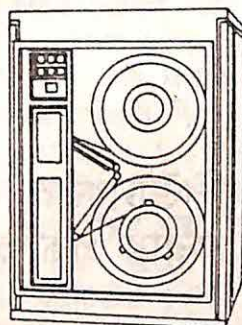
ফ্লপি ডিস্‌কটির পাশের দিকে একটি খাঁজ আছে। যখন কোনো ফ্লপিতে এই খাঁজটি ঢাকা থাকে তখন সেই ফ্লপিতে কিছু লেখা যায় না, কেবলমাত্র সেখানে যা লেখা আছে তা কমপিউটারে নিয়ে আসা চলে। ফ্লপি অনেক সময়েই বইয়ের পাতার মধ্যে নিয়ে চলাফেরা করা যায়। তবে সেখানে যেন না চাপ পড়ে, খেয়াল রাখা দরকার। আজকাল বিদেশী অনেক বইএর সঙ্গেও ফ্লপি দিয়ে দেয়। ওই বইতে যে সব প্রোগ্রাম তৈরি করা হয়েছে তা ওই ফ্লপিতে রাখা থাকে।

একটি PC-এর সঙ্গে সাধারণত দুটি করে ফ্লপি-ড্রাইভ দেওয়া হয়। আজকাল এতে 1.2M অক্ষর পর্যন্ত রাখা সম্ভব। 'M' অর্থাৎ 1 মিলিয়ন বা 10 লক্ষ। একটি PC এবং PC/XT-এর মধ্যে তফাত,

৯ সংখ্যক চিত্র ঃ একটি ফ্লপি ডিস্ক

PC/XT-এর সঙ্গে ফ্লপি ড্রাইভ ছাড়াও 'উইনচেষ্টার ডিস্ক' ড্রাইভও থাকা সম্ভব। উইনচেষ্টারে 20 M থেকে 40 M অক্ষর রাখা যায়।

যে কোনো ধরনের ডিস্ক থেকেই কিছু পড়ার সময় চৌম্বক ক্ষেত্রকে বৈদ্যুতিক সংকেতে রূপান্তর ঘটানো হয় আবার ডিস্কে কিছু লেখার সময়ে উল্টো দিক দিয়ে বৈদ্যুতিক সংকেতকে চৌম্বক ক্ষেত্রে পরিবর্তন করা হয়ে থাকে। একটি ডিস্কে যে কোনো স্থানে তথ্য রাখা এবং এর থেকে তথ্য পড়া সম্ভব বলে একে ডাইরেক্ট-অ্যাকসেস-স্টোরেজ ডিভাইস (Direct Access Storage Device) বলা হয়।

5. ম্যাগনেটিক্ টেপ (Magnetic Tape)

ডিস্কের মত টেপও অনুপ্রবেশ এবং নির্গম – উভয় অংশ হিসেবেই ব্যবহৃত হয়। তবে ডিস্কের যে কোনো অংশের সঞ্চিত তথ্য যেমন সরাসরি নিয়ে আসা সম্ভব কিংবা কোনো একটি বিশেষ অংশে তথ্য যেমন সরাসরি সঞ্চয় করা যায়, টেপে সেরকম চলে না। টেপের কোনো একটি বিশেষ অংশের তথ্য নিয়ে কাজ করতে হলে তার আগের সমস্ত তথ্য পড়ে নিয়ে তবেই সেই অংশের তথ্য আনা সম্ভব। ব্যাপারটা অনেকটা ক্যাসেটের টেপের মত। ক্যাসেটের টেপের বেলায় আমরা কি করি? একটি টেপে অনেকগুলি গান থাকলে, যদি তৃতীয় গানটি শুনতে চাই, তাহলে টেপটি ঘুরিয়ে প্রথম দুটি গানের জায়গা বাদ দিয়ে তবেই তৃতীয় গানের জায়গায় আসি। কিন্তু রেকর্ডে অনেকগুলি গান থাকলেও কোনো একটি বিশেষ গান শোনার ইচ্ছে হলে সরাসরি সেই গানের জায়গাতে চলে যাওয়া সম্ভব। তবে ডিস্কের মত ম্যাগনেটিক টেপেও চৌম্বক পদার্থের সাহায্যে যে কোনো তথ্য সঞ্চয় করা সম্ভব হয়। অবশ্য ডিস্কে দু দিকেই চৌম্বক পদার্থ থাকতে পারে। কিন্তু টেপে সাধারণত একদিকে চৌম্বক পদার্থ থাকে। সাধারণত একটি টেপ প্রস্থে 1.27 সেমি এবং দৈর্ঘ্যে 731.5 মি। এই টেপ 25.4 সেমি ব্যাসের রীলের চারপাশে জড়ানো থাকে। এই রীলের পিছনে একটি প্ল্যাসটিকের রিং লাগানোর জায়গা আছে। ফ্লপিতে যেমন খাঁজের জায়গাটি ঢাকা থাকলে সেই ফ্লপিতে কিছু লেখা যায় না এই টেপও তেমনি

১০ সংখ্যক চিত্র ঃ একটি ম্যাগনেটিক টেপ

একটি রিং সমেত হলে কমপিউটারকে দিয়ে তাতে কোনো তথ্য সঞ্চয় করা সম্ভব নয়। কেবলমাত্র টেপটিতে যে তথ্য আছে তা কমপিউটারে নিয়ে এসে কাজ করা যায়। অথচ টেপটি যদি রিং বর্জিত হয় তবেই টেপে কিছু তথ্য সঞ্চয় করা সম্ভব। গানের রেকর্ড অথবা ক্যাসেটের টেপ থেকে কোনো গান শুনতে হলে যেমন একটি বিশেষ যন্ত্রের দরকার, তেমনি এই ম্যাগনেটিক্ টেপ দিয়ে কিছু করতে হলে টেপটিকে একটি যন্ত্রের সাহায্য নিতে হয়। এই যন্ত্রটিকে টেপ-ড্রাইভ বলে। এই টেপ-ড্রাইভের উপর নির্ভর করে 1 সেমি টেপে 80 থেকে 2500টি পর্যন্ত অক্ষর সঞ্চয় করে রাখা চলে। টেপটির গতিও নিয়ন্ত্রিত করে এই টেপ-ড্রাইভটি। এই গতি সাধারণত সেকেণ্ডে 85 সেমি থেকে 508 সেমি পর্যন্ত হতে পারে। এই টেপ পরিবহনযোগ্য। কোনো একটি টেপের সঞ্চিত তথ্য অন্য একটি সংস্থায় নিয়ে গিয়ে কাজে লাগানো যায় কিন্তু সে ক্ষেত্রে দু জায়গার টেপ্-ড্রাইভ একই ধরনের হওয়া দরকার।

প্রথম দিকে সহায়ক স্মৃতি হিসেবে একমাত্র টেপেরই ব্যবহার হত। তবে আজকাল ডিস্কের ব্যবহার বেশি হয়ে থাকে। অবশ্য টেপের দাম অনেক কম, রাখার সুবিধে এবং সঙ্গে নিয়ে চলাফেরা করা যায় বলে এখনও এর ব্যবহার আছে।

কমপিউটারের
কাজ করার পদ্ধতি

কমপিউটার কিভাবে কোনো সংখ্যা বুঝতে পারে ?

কমপিউটারে সব সংখ্যাই 0 এবং 1 এর সাহায্যে বোঝানো হয়। তা ছাড়া কোনো একটি সংখ্যার প্রত্যেকটি অঙ্কের মান বের করার জন্য তিনটি তথ্য জানা প্রয়োজন। 1. সেই অঙ্কটির নিজের মান 2. সংখ্যাটিতে সেই অঙ্কটির অবস্থিতি 3. ওই সংখ্যা-পদ্ধতির ভিত্তি (base)। উদাহরণস্বরূপ, দশমিক 3562 সংখ্যাটিতে 5 অঙ্কটির কথা ধরা যাক। এই অঙ্কটির নিজের মান 5, এর অবস্থিতি তৃতীয় (অর্থাৎ শতকের ঘরে) এবং এর ভিত্তি 10। কারণ এটি একটি দশমিক সংখ্যা। কাজেই এই তিনটি তথ্যের উপরে ভিত্তি করে বলা যায়, এক্ষেত্রে 5টি শতক আছে অর্থাৎ গাণিতিক নির্দেশে তা হল 5×100 বা 5×10^2। এইভাবে 3562 দশমিক সংখ্যাটিতে 3টি হাজার, 5টি শতক, 6টি দশক এবং 2টি একক আছে। অর্থাৎ 3562 সংখ্যাটিকে লেখা যেতে পারে $3 \times 10^3 + 5 \times 10^2 + 6 \times 10^1 + 2 \times 10^0$। তাহলে দেখা যাচ্ছে, কোনো একটি দশমিক সংখ্যাকে প্রকাশের জন্য এককের ঘর, দশকের ঘর, শতকের ঘর, হাজারের ঘর অর্থাৎ 10^0, 10^1, 10^2, 10^3 প্রভৃতির ঘর রয়েছে। এখানে প্রত্যেকটি স্থানকেই 10-এর ঘাত হিসেবে দেখানো হয়। দশমিক সংখ্যায় যেহেতু দশটি অঙ্ক অর্থাৎ 0, 1, 2, 3, 4, 5, 6, 7, 8 এবং 9 ব্যবহার হয়, কাজেই এই সংখ্যা-পদ্ধতিতে 10-এর উপর ভিত্তি করে এককের (অর্থাৎ 10^0), দশকের (অর্থাৎ 10^1), শতকের (অর্থাৎ 10^2) ঘর হিসেবে লেখা হয়। অনুরূপভাবে যে সংখ্যা-পদ্ধতিতে 0 এবং 1-কেবলমাত্র এই দুটি অঙ্ক আছে, সেই সংখ্যা-পদ্ধতিতে কোনো সংখ্যা প্রকাশ করতে হলে একের ঘর (অর্থাৎ 2^0), দুয়ের ঘর (অর্থাৎ 2^1), চারের ঘর (অর্থাৎ 2^2) প্রভৃতির সাহায্যে করা সম্ভব। 0 এবং 1-যুক্ত সংখ্যা-পদ্ধতি কেবলমাত্র 2টি অঙ্কযুক্ত বলে, এই

সংখ্যা-পদ্ধতিকে দ্বি-নিধানী সংখ্যা-পদ্ধতি (Binary number system) বলে। এক্ষেত্রে 2-কে ভিত্তি হিসেবে ধরা হয়। কোনো দ্বি-নিধানি সংখ্যার সমতুল দশমিক সংখ্যা কত ? 1011 একটি দ্বি-নিধানী সংখ্যা। এর সমতুল দশমিক সংখ্যাটি বের করতে হলে দ্বি-নিধানী সংখ্যার একক, দশক, শতক, সহস্রের অঙ্কগুলি নিয়ে লেখা যেতে পারে $1 \times 2^0 + 1 \times 2^1 + 0 \times 2^2 + 1 \times 2^3$ অর্থাৎ $1 + 2 + 0 + 8$ বা 11_{10}। সুতরাং দেখা যাচ্ছে 1011 এই দ্বি-নিধানী সংখ্যাটির সমান দশমিক সংখ্যা 11_{10}। এখানে একটা কথা উল্লেখ করা দরকার, একটি সংখ্যা কোন সংখ্যা পদ্ধতিতে প্রকাশ করা হয়েছে তা বোঝানোর জন্যে সেই সংখ্যাটির নীচে সংখ্যা-পদ্ধতিটির ভিত্তি উল্লেখ করা হয়, যেমন, 1011_2। এই সংখ্যাটির নীচে 2 থাকায় বোঝা যাচ্ছে, সংখ্যাটি দ্বি-নিধানী সংখ্যা পদ্ধতিতে লেখা হয়েছে।

কোনো একটি দশমিক সংখ্যার সমতুল দ্বি-নিধানী সংখ্যা বের করাও সহজ। এক্ষেত্রে ভাগফল 1-না আসা পর্যন্ত সংখ্যাটিকে ক্রমান্বয়ে 2 দিয়ে ভাগ করে যেতে হবে। আর ভাগশেষগুলি নিয়েই দ্বি-নিধানী সংখ্যাটি তৈরি হবে। তাহলে 19_{10} দশমিক সংখ্যার সমান দ্বি-নিধানী সংখ্যাটি কত হবে ?

		ভাগশেষ
2	19	
2	9	1
2	4	1
2	2	0
	1	0

তা হল 10011_2। এইভাবে যে কোনো দশমিক সংখ্যাকেই দ্বি-নিধানী সংখ্যা হিসেবে লেখা সম্ভব। উপরের উদাহরণে, দশমিক সংখ্যা 19_{10}-কে লিখতে দশমিক পদ্ধতিতে দুটি অঙ্ক প্রয়োজন, কিন্তু সেই একই সংখ্যা দ্বি-নিধানী সংখ্যা-পদ্ধতিতে লিখবার সময়ে পাঁচটি অঙ্কের দরকার হচ্ছে। এখন কমপিউটারে একটি বিট দিয়ে 0 কিংবা 1 বোঝানো সম্ভব। কাজেই একটি দ্বি-নিধানী অঙ্ক 0 বা 1-এর জন্য কমপিউটারে একটি বিটের প্রয়োজন। 19_{10} সংখ্যাটির সমতুল সংখ্যা 10011_2 তে পাঁচটি বিট লাগবে।

অবশ্য এখানে একটা কথা বলা যায় যে, 1টি বিট দিয়ে দশমিক সংখ্যা 0_{10} এবং 1_{10} বোঝানো চলে। এইভাবে 2টি বিট দিয়ে চারটি দশমিক সংখ্যা লেখা যায়। যেমন, $0_{10}(00_2)$, $1_{10}(01_2)$, $2_{10}(10_2)$ এবং $3_{10}(11_2)$। তাহলে 3-টি বিট দিয়ে কতগুলি দ্বি-নিধানী সংখ্যা এবং কতগুলিই বা দশমিক সংখ্যা লেখা সম্ভব ? তিনটি মাত্র বিট দিয়ে

যে সব দ্বি-নিধানী সংখ্যা লেখা যায় সেই সব সংখ্যা এবং তাদের সমতুল দশমিক সংখ্যার একটি তালিকা দেওয়া হল।

3-বিট দ্বি-নিধানী সংখ্যা	দশমিক সংখ্যা
000 $(= 0 \times 2^0 + 0 \times 2^1 + 0 \times 2^2)$	0
001 $(= 1 \times 2^0 + 0 \times 2^1 + 0 \times 2^2)$	1
010 $(= 0 \times 2^0 + 1 \times 2^1 + 0 \times 2^2)$	2
011 $(= 1 \times 2^0 + 1 \times 2^1 + 0 \times 2^2)$	3
100 $(= 0 \times 2^0 + 0 \times 2^1 + 1 \times 2^2)$	4
101 $(= 1 \times 2^0 + 0 \times 2^1 + 1 \times 2^2)$	5
110 $(= 0 \times 2^0 + 1 \times 2^1 + 1 \times 2^2)$	6
111 $(= 1 \times 2^0 + 1 \times 2^1 + 1 \times 2^2)$	7

এই তালিকা থেকে বলা যেতে পারে, 0 থেকে 7 পর্যন্ত যে কোনো আটটি দশমিক সংখ্যা 3টি বিটের সাহায্যে প্রকাশ করা চলে। দশমিকে কতগুলি সংখ্যা পাওয়া যাবে, বিটের সংখ্যার সঙ্গে তার একটা সম্পর্ক আছে। তিনটি বিটের বেলায় প্রাপ্ত সংখ্যা 8 অর্থাৎ 2^3। এইভাবে চারটি বিটের বেলায় প্রাপ্ত সংখ্যা হবে 2^4 অর্থাৎ 16, পাঁচের বেলায় 2^5 অর্থাৎ 32। যদিও তিনটি বিট দিয়ে মোট 8টি সংখ্যা পাওয়া যায়, কিন্তু তিন বিটের সবচেয়ে বড় দশমিক সংখ্যা হয় 7 অর্থাৎ $2^3 - 1$। এইভাবে চারটি বিটে সবচেয়ে বড় দশমিক সংখ্যা 15 (অর্থাৎ $2^4 - 1$), পাঁচটি বিটে 31 (অর্থাৎ $2^5 - 1$)। অনুরূপভাবে একটি কমপিউটারে যদি প্রত্যেকটি কোষে 16টি করে বিট থাকে তবে সেই কমপিউটারে সবচেয়ে বড় যে দশমিক সংখ্যাটি সঞ্চয় করা যেতে পারে তা হল $2^{16} - 1$ অর্থাৎ 65535। এর চাইতে বড় কোনো দশমিক সংখ্যা ওই কমপিউটারে রাখা সম্ভব নয়। তবে 16 বিটের কমপি-উটারে এত বড় সংখ্যাও রাখা যায় না, তার কারণ কমপিউটারে শুধুমাত্র ধনাত্মক সংখ্যা দিয়েই কাজ হয় না, সেখানে ঋণাত্মক সংখ্যারও দরকার আছে। কাজেই এই দুই ধরনের সংখ্যাই বোঝানোর ব্যবস্থা কমপিউটারে করা থাকে। একটি 16টি বিটের কমপিউটারে একটি বিটকে নির্দিষ্ট রাখা হয় সংখ্যাটি ধনাত্মক না ঋণাত্মক তা বোঝানোর জন্য। এজন্যে সাধারণত প্রথম কিংবা শেষ বিটটিকে ধরে নেওয়া হয়। মনে করা যাক, 16 বিটের কমপিউটারে 16 সংখ্যক বিটটি নির্দিষ্ট রাখা আছে। এক্ষেত্রে কোনো সংখ্যা ধনাত্মক বোঝাতে 16 সংখ্যক বিটটিতে 0 এবং ঋণাত্মক বোঝাতে 1 থাকবে। এখন একটি কোষে একটি বিট ধনাত্মক এবং ঋণাত্মক বোঝানোর জন্য নির্দিষ্ট থাকায় বাকি কেবলমাত্র 15টি বিটই পাওয়া যাবে সংখ্যা লেখার জন্য। কাজেই কোনো 16 বিট কমপিউটারে

সবচেয়ে বড় ধনাত্মক সংখ্যা $2^{15} - 1$ অর্থাৎ 32767 এবং কখনোই এর চেয়ে বেশি হওয়া সম্ভব নয়।

এখন সংখ্যা ছোট-বড় যাই হোক, একটি দশমিক পূর্ণ সংখ্যা কিভাবে দ্বি-নিধানী সংখ্যায় লেখা হয় বোঝা গেল, কিন্তু একটি দশমিক ভগ্নাংশ ওই নূতন সংখ্যা-পদ্ধতিতে লেখা হবে কেমন করে ?

একটি দশমিক পূর্ণ সংখ্যাকে এককের ঘর, দশকের ঘর, শতকের ঘর প্রভৃতি অবস্থানের সাহায্যে প্রকাশ করা হয়। 3562 সংখ্যাটিকে কি ভাবে লেখা হবে ? দশমিক পদ্ধতিতে ওই সংখ্যাটি লেখা যায় $3 \times 10^3 + 5 \times 10^2 + 6 \times 10^1 + 2 \times 10^0$। অনুরূপভাবে, দশমিক ভগ্নাংশ লিখতে $1/_{10}$ অর্থাৎ 10^{-1}, $1/_{10^2}$ অর্থাৎ 10^{-2}, $1/_{10^3}$ অর্থাৎ 10^{-3} ইত্যাদি অবস্থানের সাহায্য নেওয়া হয়। তহলে $4 \times 10^{-1} + 7 \times 10^{-2} + 8 \times 10^{-3} + 9 \times 10^{-4}$ হচ্ছে .4789। ঠিক একই ভাবে দ্বি-নিধানী সংখ্যা-পদ্ধতিতেও এগোনো চলে। সেখানে .1011 লেখা যাবে $1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3} + 1 \times 2^{-4}$। তাহলে যে কোনো দশমিক সংখ্যা দ্বি-নিধানী সংখ্যা-পদ্ধতিতে নীচের সূত্র অনুযায়ী লেখা সম্ভব।

$$d = p_n \,.\, 2^n + p_{n-1} \,.\, 2^{n-1} + \ldots\ldots + p_1 \,.\, 2^1 + p_0 \,.\, 2^0$$
$$+ p_{-1} \,.\, 2^{-1} + p_{-2} \,.\, 2^{-2} + \ldots\ldots + p_{-m} \,.\, 2^{-m}$$

এখানে $p_n, p_{n-1}, \ldots, p_1, p_0, p_{-1}, p_{-2}, \ldots, p_{-m}$ এইগুলি এক একটি দ্বি-নিধানী সংখ্যা। d তে পূর্ণ এবং ভগ্নাংশ ওই দু ধরনের সংখ্যাই থাকা সম্ভব। পূর্ণ সংখ্যা i দিয়ে এবং ভগ্নাংশ f দিয়ে লিখলে উপরের সূত্রটি হবে

$$d = i \cdot f = p_n \,.\, 2^n + p_{n-1} \,.\, 2^{n-1} + \ldots + p_1 \,.\, 2^1 + p_0 \,.\, 2^0$$
$$+ p_{-1} \,.\, 2^{-1} + p_{-2} \,.\, 2^{-2} + \ldots + p_{-m} \,.\, 2^{-m} .$$

অর্থাৎ $i = p_n \,.\, 2^n + p_{n-1} \,.\, 2^{n-1} + \ldots + p_1 \,.\, 2^1 + p_o \,.\, 2^0$

এবং $\cdot f = p_{-1} \,.\, 2^{-1} + p_{-2} \,.\, 2^{-2} + \ldots + p_{-m} \,.\, 2^{-m}.$

i দশমিক পূর্ণ সংখ্যাটিকে দ্বি-নিধানী সংখ্যায় লিখতে হলে i একে ক্রমান্বয়ে 2 দিয়ে ভাগ করে যেতে হবে। প্রথমবার ভাগ করলে যে ভাগশেষ পাওয়া যাবে সেটিই হবে p_0। ভাগফলটিকে আবার 2 দিয়ে ভাগ করে p_1 পাওয়া যাবে। এইভাবে ক্রমান্বয়ে ভাগ করলে সর্বশেষ ভাগশেষ p_n। আবার p_{-1}, p_{-2} প্রভৃতি বের করতে হলে .f কে যথাক্রমে 2 দিয়ে গুণ করে যেতে হবে। প্রথমবার. f কে 2 দিয়ে গুণ করলে যে অখণ্ড সংখ্যাটি পাওয়া যাবে সেটিই হবে p_{-1}। পরের অখণ্ড সংখ্যাটি কত ? p_{-1} পাওয়ার পরে যে অবশিষ্ট দশমিক ভগ্নাংশটি থেকে যায়, তাকে আবার 2 দিয়ে গুণ করে প্রাপ্ত অখণ্ড সংখ্যাটি p_{-2}। এইভাবে ক্রমান্বয়ে 2 দিয়ে গুণ করে শেষ পর্যন্ত p_{-m} পাওয়া যাবে। 2 দিয়ে ক্রমান্বয়ে গুণ কতক্ষণ চলবে ? যতক্ষণ পর্যন্ত না ভগ্নাংশটি শূন্য আসে

অথবা একটি স্মৃতি কোষে ভগ্নাংশের জন্য নির্দিষ্ট রাখা সম্ভব সব কটি বিট যতক্ষণ পর্যন্ত না বের করা সম্ভব হয়, ততক্ষণ পর্যন্ত গুণের কাজ চালিয়ে যেতে হবে। এখানে মনে রাখতে হবে, ক্রমান্বয়ে গুণ করেও যদি শেষ পর্যন্ত শূন্য না হয় তাহলে যে দ্বি-নিধানী সংখ্যাটি পাওয়া যায় সেটি দশমিক ভগ্নাংশটির সঙ্গে অভিন্ন থাকবে। দশমিকের পরে কয়েকটি স্থানের পর তা আর সমান হবে না। অর্থাৎ এক্ষেত্রে সংখ্যাটিকে ঠিকমত কমপিউটারে সঞ্চয় করা গেল না, একটু ভুল থেকে গেল। উদাহরণ নিয়ে দেখা যাক কি ভাবে একটি দশমিক ভগ্নাংশকে দ্বি-নিধানী সংখ্যায় লেখা যায়ঃ

দশমিক ভগ্নাংশ 0.75 দ্বি-নিধানী ভগ্নাংশে কত হবে?

$0.75 \times 2 = 1.50$ → 1
$0.50 \times 2 = 1.00$ → 1

কাজেই দশমিক ভগ্নাংশ 0.75 হচ্ছে দ্বি-নিধানী ভগ্নাংশ 0.11-এর সমান। এখানে দ্বিতীয়বার গুণ করার পর গুণের কাজ সম্পূর্ণ হয়েছে, কারণ সেক্ষেত্রে অবশিষ্ট ভগ্নাংশটি 0 হয়ে গেছে। অনুরূপভাবে দশমিক ভগ্নাংশ 0.25 দ্বি-নিধানী ভগ্নাংশে হবে 0.01. কারণ,

$0.25 \times 2 = 0.50$ → 0
$0.50 \times 2 = 1.00$ → 1

এবারে দেখা যাক, 0.31 এই দশমিক সংখ্যাটি কিভাবে দ্বি-নিধানী সংখ্যাতে লেখা যাবে।

$0.31 \times 2 = 0.62$ → 0
$0.62 \times 2 = 1.24$ → 1
$0.24 \times 2 = 0.48$ → 0
$0.48 \times 2 = 0.96$ → 0
$0.96 \times 2 = 1.92$ → 1
$0.92 \times 2 = 1.84$ → 1
$0.84 \times 2 = 1.68$ → 1
$0.68 \times 2 = 1.36$ → 1

এখন মনে করা যাক, কমপিউটারে দ্বি-নিধানী ভগ্নাংশটির জন্য ৪টি বিট নির্দিষ্ট করা আছে। কাজেই 0.31_{10}- এর সমান দ্বি-নিধানী সংখ্যাটি হবে 0.01001111_2। কিন্তু সত্যিই কি 0.01001111_2 এই সংখ্যাটি একেবারে 0.31_{10} এর সমান ?

$0.01001111 = 0x2^{-1} + 1x2^{-2} = 0x2^{-3} + 0x2^{-4} + 1x2^{-5} + 1x2^{-6} + 1x2^{-7} + 1x2^{-8}$

$= 0 + {}^1/_4 + 0 + 0 + {}^1/_{32} + {}^1/_{64} + {}^1/_{128} + {}^1/_{256}$
$= 0.25 + 0.03125 + 0.015625 + 0.0078125$
$+ 0.00390625$
$= 0.30859375$

অর্থাৎ 0.01001111_2 একেবারে 0.31-এর সমান নয়, কিন্তু তার কাছাকাছি। এর থেকে আরও কাছাকাছি আসতো যদি ওই কমপিউটারে কিছু বেশি সংখ্যক বিট ভগ্নাংশের জন্য নির্দিষ্ট থাকতো। এক্ষেত্রে ৪টি বিট নির্দিষ্ট থাকায় ৪টি বিটের সাহায্যে যতটা কাছাকাছি পাওয়া সম্ভব তাই রাখা হয়েছে। সুতরাং কমপিউটারে একটি স্মৃতি কোষে সীমিত সংখ্যক বিট থাকে বলে দশমিক সব ভগ্নাংশের সমানই দ্বি-নিধানী সংখ্যা বের করা সম্ভব নাও হতে পারে। কাজেই ক্রমান্বয়ে গুণ করে এগোনোর ক্ষেত্রে এক সময়ে আমাদের থামতেই হয় এবং সেই সময়েও অবশিষ্ট ভগ্নাংশটি শূন্য না হওয়ার সম্ভাবনা রয়ে যায়। এখানে একটা কথা বলা দরকার, যে কোনো দশমিক পূর্ণ সংখ্যাই কমপিউটারে সঞ্চয় করা সম্ভব নয়। সবচেয়ে বড় কোন দশমিক পূর্ণ সংখ্যা কমপিউটারে সঞ্চয় করা যাবে তা ওই কমপিউটারের স্মৃতি কোষের বিটের সংখ্যার উপরে সম্পূর্ণভাবে নির্ভর করে। কমপিউটারে একটি কোষে সব সময়েই সীমিত সংখ্যক বিট থাকে। দশমিক পূর্ণ সংখ্যার বেলাতে একটি সংখ্যা উল্লেখ করে বলা যায়, এর চেয়ে বড় কোনো সংখ্যা রাখা সম্ভব নয়। কিন্তু দশমিক ভগ্নাংশের বেলায় তা বলার উপায় নেই। তখন কেবল এটুকুমাত্র বলা যায় যে, ক্রমান্বয়ে গুণ করে যদি অবশিষ্ট ভগ্নাংশ শূন্য হয় তাহলে সেই ভগ্নাংশটি সঠিকভাবে কমপিউটারে রাখা যাবে, আর তা না হলে ওই ভগ্নাংশটির খুব কাছাকাছি একটি সংখ্যা কমপিউটারে সঞ্চিত হবে। তখন ফলাফলে কিছুটা ভুল হলেও হতে পারে। এক্ষেত্রে যে মান পাওয়া যাবে তা হবে আসন্ন মান।

কমপিউটারের সাহায্যে পাটীগণিত ঃ

এই দ্বি-নিধানী সংখ্যা-পদ্ধতিতে কি ভাবে যোগ, বিয়োগ, গুণ ও ভাগ করা হয়? আমরা জানি, কমপিউটারে ইলেকট্রনিক সার্কিটের সাহায্যে সব কাজ চলে এবং যে কোনো দুটি দ্বি-নিধানী সংখ্যা যোগ করতে ইলেকট্রনিক সার্কিটে 'লজিক গেট' (Logic gates)-এর সাহায্য নেওয়া হয়। 'লজিক গেট' বিষয়টি কি? 'লজিক গেট' বুঝতে গেলে আবার 'বুলিয়ান' (Boolean) বীজগণিত সম্বন্ধে কিছু জানা দরকার। সাধারণ বীজগণিতে যেমন +, – প্রভৃতি চিহ্ন ব্যবহার করে বিভিন্ন ধরনের প্রক্রিয়া বোঝানো হয়, বুলিয়ান

বীজগণিতেও সেরকম 'এ্যাণ্ড' (AND), 'অর' (OR), 'নট' (NOT) ব্যবহার করে বিভিন্ন ধরনের কাজ বোঝানো হয়ে থাকে। তবে বুলিয়ানে একটি সহজ সরল উক্তির বদলে একটি প্রতীক ব্যবহার করা হয়। 'অমলের কমপিউটার আছে' এই উক্তির ক্ষেত্রে কি হবে ? এখানে উক্তিটির বদলে একটি প্রতীক A ব্যবহার করা যেতে পারে। এখন A সত্য অথবা অসত্য হওয়া সম্ভব। অর্থ্যাৎ অমলের কমপিউটার থাকলে A সত্য বলে ধরা হবে, না থাকলে A অসত্য মনে করা হবে। ওই রকম আর একটি উক্তির বদলে B লেখা চলে। B- ও A-এর মত সত্য অথবা অসত্য হতে পারে। কমপিউটারে সত্যকে 1 দিয়ে এবং অসত্যকে 0 দিয়ে বোঝানো সম্ভব। A এবং B কে বুলিয়ানে 'এ্যাণ্ড' করলে একটি জটিল উক্তির সৃষ্টি হবে এবং সেই উক্তিটিকে C দিয়ে নির্দেশ করলে C কোনো কোনো ক্ষেত্রে 1 হবে আবার কোনো কোনো ক্ষেত্রে 0 । কোনো ক্ষেত্রে 1 আবার কোনো কোনো ক্ষেত্রে 0 হবে এখানে তা দেখানো হচ্ছে।

A	এ্যাণ্ড	B	=	C
1		1		1
1		0		0
0		1		0
0		0		0

উপরের এ্যাণ্ড টেবিল থেকে বোঝা যাচ্ছে A এবং B দুটিই যখন 1 তখনই C 1 হবে এবং আর সব ক্ষেত্রেই C 0 হবে। অর্থাৎ দুটি উক্তিই যখন সত্য কেবলমাত্র তখনই C সত্য হবে এবং C 1 হবে।

A এবং B- কে বুলিয়ানে 'অর' করলে C কখন কি হবে তা এবারে দেখানো যাক।

A	অর	B	=	C
1		1		1
1		0		1
0		1		1
0		0		0

এক্ষেত্রে A এবং B দুটিই 0 হলে তবেই C 0 হবে আর সব সময়েই C 1 হবে। অর্থাৎ এক্ষেত্রে A এবং B-এর যে কোনো একটি বা উভয়ই সত্য হলে C সত্য হবে। আর কেবলমাত্র A এবং B দুটিই অসত্য হলে তবেই C অসত্য হবে।

এখানে একথা বলা প্রয়োজন যে, 'এ্যাণ্ড' অথবা 'অর' করতে হলে স্বাভাবিক ভাবেই দুটি উক্তির প্রয়োজন। কিন্তু 'নট'-এর

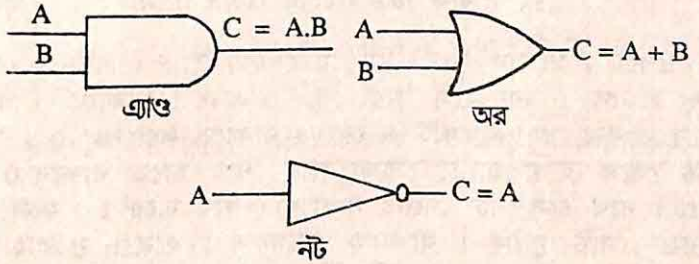
বৈশিষ্ট্য, কেবলমাত্র একটি উক্তির আগেই লাগানো যায়। C যদি 'নট' A হয় তবে A 1 হলে, C 0 হবে এবং A 0 হলে, C 1 হবে।

যে সব ইলেকট্রনিক সার্কিট 'এ্যাণ্ড', 'অর' অথবা 'না' এই কাজগুলি করতে পারে তাদরই 'লজিক গেট' বলা হয়। এই তিনটি 'লজিক গেট'-এর সাহায্যে আবার বর্তনী বা সার্কিট তৈরি করে কমপিউটারে দুটি দ্বি-নিধানী সংখ্যা যোগ করা সম্ভব হয়।

এই লজিক গেট তিনটির ছবি 11 সংখ্যক চিত্রে এবং দুটি দ্বি-নিধানী সংখ্যা যোগ করার একটি সার্কিট 12 সংখ্যক চিত্রে দেখানো হচ্ছে –

A B C = A.B এ্যাণ্ড

A B C = A + B অর

A C = A নট

১১ সংখ্যক চিত্র ঃ লজিক গেট

এর ভিতর এ্যাণ্ড -এর ছবিটিতে দেখা যাচ্ছে A এবং B- কে 'এ্যাণ্ড' করলে A এবং B- এর মধ্যবর্তী স্থানে ডট (.) বসিয়ে তা বোঝানো হয়। A এবং B অনুরূপভাবে 'অর'-এর বেলায় যোগ চিহ্নের সাহায্যে এবং 'নট'-এ A এবং B- এর উপর মাত্রা দিয়ে তা বোঝানো হয়ে থাকে।

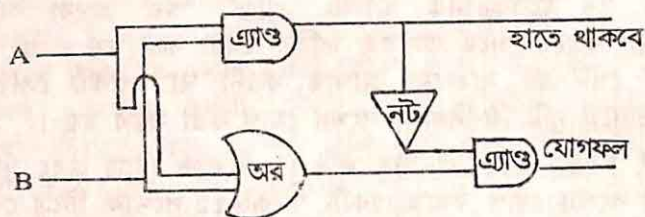
কমপিউটারে যোগ করার জন্য নীচের চারটি সূত্রকে মনে রেখে ইলেকট্রনিক সার্কিট তৈরি করলেই যে কোনো দুটি দ্বি-নিধানী সংখ্যা যোগ করা যাবে।

1. 0- এর সঙ্গে 0 যোগ করলে যোগফল 0 হবে এবং হাতে 0 থাকবে।

2. 0- এর সঙ্গে 1 যোগ করলে যোগফল 1 হবে এবং হাতে 0 থাকবে।

3. 1- এর সঙ্গে 0 যোগ করলে যোগফল 1 হবে এবং হাতে 0 থাকবে।

4. 1- এর সঙ্গে 1 যোগ করলে যোগফল 0 হবে এবং হাতে 1 থাকবে।

12 সংখ্যক চিত্রের সার্কিটটির সাহায্যে এই চারটি সূত্র করাই সম্ভব।

১২ সংখ্যক চিত্র ঃ যোগ করার বর্তনী

প্রথমে 2 সংখ্যক সূত্রটি মিলিয়ে নেওয়া যাক। এখন A-তে 0 এবং B-তে 1 ধরা হলে 'অর' গেটে 0 এবং 1 থাকাতে 1 পাওয়া যাবে আবার 'অ্যাণ্ড' গেটে 0 এবং 1 থাকাতে লব্ধ ফল 0। 'এ্যাণ্ড' গেট থেকে 0 পাওয়াতে, সোজা পথে গিয়ে হাতে থাকবে 0 এবং নীচের পথে এসে 'নট' গেটের সাহায্যে 0 হয়ে যাবে 1। ফলে পরের 'এ্যাণ্ড' গেটে দুটিই 1 থাকাতে যোগফল 1 হবে। সুতরাং দেখা যাচ্ছে, এস্থলে 2 সংখ্যক সূত্রটি ঠিকমত পাওয়া যাচ্ছে। এরপর 4-সংখ্যক সূত্র অনুযায়ী, ধরা যাক, A-তে 1 এবং B-তে 1 আছে। অতএব 'অর' এবং 'এ্যাণ্ড' গেট দিয়ে 1 পাওয়া যাবে। উপরের 'এ্যাণ্ড' গেট দিয়ে 1 পাওয়াতে হাতে থাকবে 1 এবং 1 'নট' গেটের সাহায্যে 0 হবে। পরের 'অর' গেট দিয়ে 1 পাওয়া গেছে এবং এই 'নট' গেট থেকে পাওয়া 0 দিয়ে পরের 'এ্যাণ্ড' গেট দিয়ে যোগফল 0 পাওয়া যাবে। ফলে 4-সংখ্যক সূত্রটিও ঠিকমত মিলিয়ে নেওয়া গেল। অন্য সূত্র দুটিও এইভাবে এই সার্কিট দিয়ে সঠিক মিলিয়ে নেওয়া সম্ভব।

যোগের চারটি সূত্রের সাহায্যে এবারে দুটি দ্বি-নিধানী সংখ্যা যোগ করে দেখানো যেতে পারে।

$$\begin{array}{r} 1011_2 \ (= 11_{10}) \\ +\ 1101_2 \ (= 13_{10}) \\ \hline 11000_2 \ (=24_{10}) \end{array}$$

4 সংখ্যক সূত্র অনুসারে 1 এর সঙ্গে 1 যোগ করলে ফল 0 এবং হাতে 1 থাকে। এর পর 1 এবং 0 যোগ করলে 3 সংখ্যক সূত্র অনুযায়ী ফল 1। কিন্তু হাতে 1 থাকাতে সেটি যোগ হয়ে আবার 0 হবে এবং হাতে 1 থাকবে। পরের ঘরে আবার 0 এবং 1 যোগ করে 1 হবে এবং হাতের 1-এর সঙ্গে যোগের ফল আবার 0 হবে এবং হাতে 1 থাকবে। শেষের ঘরে 1-এর সঙ্গে 1 যোগ করে 0 এবং হাতে 1

থাকবে। এরপর আর কোনো সংখ্যা না থাকাতে হাতে যে 1 ছিল তা বসবে। যোগ করার সময় একটা কথা মনে রাখা দরকার, কমপিউটারে সবচেয়ে বড় যে সংখ্যা লেখা যায় দুটি সংখ্যা যোগ করার পরে যোগফল কখনোই যেন সেই সংখ্যার থেকে বড় না হয়। সে রকম হলে মুশকিল। তখন যোগফল ঠিকমত হবে না। এইভাবে কমপিউটারে যে কোনো দুটি সংখ্যা যোগ করা সম্ভব। কিন্তু বিয়োগের বেলায় কি হবে ?

একটি সংখ্যা আর একটি সংখ্যা থেকে বিয়োগ করার বেলায় কমপিউটারে যোগের সাহায্যেই তা করা হয়ে থাকে। যে নিয়ম অনুসারে তা সম্ভব প্রথমে দুটি দশমিক সংখ্যার সাহায্যে দেখানো যাক।

17-কে 33 থেকে বিয়োগ করবো। কমপিউটারে তা কি ভাবে করবো ?

সাধারণ বিয়োগ	কমপিউটারে যে নিয়মে বিয়োগ করা হয়
33_{10}	33_{10}
$-\ 17_{10}$	$+\ 82_{10}$
16_{10}	115
	$+\ 1$
	16_{10}

কমপিউটারের নিয়মে প্রথমে যে সংখ্যাটি বিয়োগ করা হবে সেই দশমিক সংখ্যা 17-এর প্রত্যেকটি অঙ্কের 9-এর প্রতিপূরক অঙ্ক বের করা হল। 9-এর প্রতিপূরক কি ভাবে বের করা হবে ? 9-এর প্রতিপূরক বের করার জন্য প্রথমে সেই অঙ্কটির সঙ্গে কত যোগ করলে 9 হবে তা বের করতে হয়। কাজেই 7-এর ক্ষেত্রে 9-এর প্রতিপূরক 2 এবং 1 এর ক্ষেত্রে তা হবে 8। সুতরাং 17 সংখ্যাটির ক্ষেত্রে 99-এর প্রতিপূরক 82। এরপর 33 এর সঙ্গে 82 যোগ করলে পাওয়া যাবে 15 এবং হাতে থাকবে 1। এখানে হাতে থাকার কথা এজন্যই বলতে হচ্ছে কারণ সংখ্যা দুটিই দুই অঙ্কের। কাজেই যোগ করার পরে যোগফল দু অঙ্কের বেশি হলেই হাতে থাকার কথা বলা হয়। এই হাতের 1-কে এখন 15-এর সঙ্গে যোগ করলে আমরা বিয়োগফল পাই 16। এখন প্রশ্ন উঠতে পারে, দুই অঙ্ক থেকে একটি এক অঙ্ক বিয়োগ করার সময়েও কি হাতে থাকার কথা বলতে হবে ? একটি উদাহরণের সাহায্যে লক্ষ্য করা যাক।

38 থেকে 9 বিয়োগ করার সময় কি ভাবে এগোতে হবে ?

সাধারণ বিয়োগ	কমপিউটারে যে নিয়মে বিয়োগ করা হয়
38_{10}	38_{10}
$-\ 9_{10}$	$+\ 90_{10}$
29_{10}	128
	+ 1
	29_{10}

এক্ষেত্রে দুই অঙ্কের সংখ্যা থেকে এক অঙ্কের সংখ্যাটিকে বিয়োগ করা হচ্ছে বলে এক অঙ্কের সংখ্যাটিকে দুই অঙ্কের সংখ্যায় প্রথমে বদলে নেওয়া হল। এটি করা খুবই সহজ। এখানে এক অঙ্কের সংখ্যাটির আগে একটি 0 বসিয়ে দিলে সংখ্যাটিকে 09 লেখা যাবে এবং তাহলেই এটি দুই অঙ্কের সংখ্যায় পরিণত হবে। এরপর আগের নিয়মে 09 সংখ্যাটির 99-এর প্রতিপূরক সংখ্যাটি হবে 90 এবং ওই সংখ্যাটি আগের নিয়মেই যোগ করে বিয়োগ করে বিয়োগফল পাওয়া যাবে।

এবারে দ্বি-নিধানী সংখ্যা-পদ্ধতিতে কি ভাবে কমপিউটারে বিয়োগ করা হয় তা উদাহরণের সাহায্যে দেখানো হবে। এক্ষেত্রে 9-এর পরিবর্তে 1-এর প্রতিপূরক বের করা হয়। 1-এর প্রতিপূরক বের করার অর্থ হল, যেখানে 1 থাকবে সেখানে 0 বসবে এবং 0 থাকলে 1 হবে। এটি 'নট' গেট দিয়ে অতি সহজেই করা যাবে। অর্থাৎ যে দ্বি-নিধানী সংখ্যাটি বিয়োগ করা দরকার, 'নট' গেটের সাহায্যে সেই সংখ্যাটির প্রতিটি অঙ্কের বিপরীত অঙ্কটি বের করে নিতে হয়। উপরের প্রথম উদাহরণটি এবারে দ্বি-নিধানী সংখ্যা-পদ্ধতিতে কি ভাবে কার্যকর হবে তা দেখা যাক। 33 থেকে 17 বিয়োগ করি আগের মতই। তাহলে প্রথমে 33_{10} এবং 17_{10} সংখ্যা দুটির সমতুল দ্বি-নিধানী সংখ্যা দুটি বের করে নিতে হবে।

		ভাগশেষ
2	33	
2	16	1
2	8	0
2	4	0
2	2	0
	1	0

		ভাগশেষ
2	17	
2	8	1
2	4	0
2	2	0
	1	0

অর্থাৎ 33_{10} এবং 17_{10} এই দুটি সংখ্যার সমতুল্য দ্বি-নিধানী সংখ্যা দুটি হল যথাক্রমে 100001_2 এবং 010001_2। যে কমপিউটারে এই

যোগটি করা হবে, ধরা যাক, সেখানে প্রত্যেকটি কোষে ৪টি করে বিট আছে। এখন কোনো একটি সংখ্যা ধনাত্মক না ঋণাত্মক বোঝানোর জন্য একটি বিটকে নির্দিষ্ট রাখা হয়। ধরে নেওয়া হচ্ছে, ওই কমপিউটারে অষ্টম বিটটি এজন্য নির্দিষ্ট আছে। তাহলে সংখ্যা দুটিকে অবশিষ্ট 7 বিটে লিখতে হবে। সুতরাং ওই দুটি সংখ্যা হবে যথাক্রমে 0100001_2 এবং 0010001_2। এবারে দ্বিতীয় সংখ্যাটিকে প্রথম সংখ্যাটি থেকে বাদ দিতে হবে বলে ওই সংখ্যাটির প্রতিটি অঙ্ককে 1-এর প্রতিপূরক হিসেবে লিখতে হবে। 1-এর প্রতিপূরক, অর্থাৎ প্রতিটি 1 0 এবং প্রতিটি 0 1 হবে। তাহলে পাওয়া যাবে 1101110_2। এখন ওই দুটি সংখ্যাকে এদের ধনাত্মক বা ঋণাত্মক বিট সমেত আটটি বিটের সাহায্যে লিখলে হবে 00100001_2 এবং 11101110_2। অষ্টম বিটটি দেখেই বোঝা যাচ্ছে প্রথম সংখ্যাটি ধনাত্মক এবং দ্বিতীয়টি ঋণাত্মক। এবারে সংখ্যা দুটিকে যোগ করার সময়ে ৪টি বিটই যোগ করা হয়। অর্থাৎ যে বিটটি সংখ্যাটি ধনাত্মক না ঋণাত্মক বোঝানোর জন্য রাখা হয়েছে সেটিও যোগের আওতায় এসে পড়ে।

$$
\begin{array}{r}
0\,0\,1\,0\,0\,0\,0\,1_2 \\
+\,1\,1\,1\;0\,1\,1\,1\,0_2 \\
\hline
1\,0\,0\,0\;0\,1\,1\,1\,1_2 \\
+ \quad \longrightarrow 1 \\
\hline
0\,0\,0\;1\,0\,0\,0\,0_2 \;(=16_{10})
\end{array}
$$

যোগফলটির অষ্টম বিটটি 0 থাকায় এটি একটি ধনাত্মক সংখ্যা।

এবারে একটি ছোট সংখ্যা থেকে কি ভাবে কমপিউটার একটি বড় সংখ্যা বিয়োগ করবে ?

মনে করা যাক, ছোট সংখ্যাটি 34_{10} এবং বড় সংখ্যাটি 48_{10}। ওই দুটি সংখ্যার সমতুল দ্বি-নিধানী সংখ্যা বের করা যাক।

		ভাগশেষ
2	34	
2	17	0
2	8	1
2	4	0
2	2	0
	1	0

		ভাগশেষ
2	48	
2	24	0
2	12	0
2	6	0
2	3	0
	1	1

তাহলে এই সংখ্যা দুটি যথাক্রমে 100010_2 এবং 110000_2। কিন্তু

এই কমপিউটারের প্রতিটি কোষে আটটি করে বিট থাকার জন্যে প্রতিটি সংখ্যাকেই সাতটি বিট দিয়ে লিখতে হবে এবং ধনাত্মক না ঋণাত্মক বোঝানোর জন্য অষ্টম বিটটি নির্দিষ্ট থাকবে। কাজেই সাতটি বিটের সাহায্যে ব্যক্ত সংখ্যা দুটি হবে 0100010 এবং 0110000। এক্ষেত্রে প্রথম সংখ্যাটি ধনাত্মক এবং দ্বিতীয় সংখ্যাটিকে প্রথমটি থেকে বিয়োগ করতে হবে। কাজেই দ্বিতীয় সংখ্যাটিকে ঋণাত্মক হিসেবে ধরে নিতে হবে। সুতরাং দ্বিতীয় সংখ্যাটির প্রত্যেকটি অঙ্কের 1-এর প্রতিপূরক বের করে তবেই সংখ্যাটিকে ঋণাত্মক হিসেবে লিখতে হবে। এখন দ্বিতীয় সংখ্যাটি 0110000_2-এর প্রতিটি অঙ্কের 1-এর প্রতিপূরক বের করলে তা হবে 1001111_2। সুতরাং সংখ্যা দুটি ধনাত্মক বা ঋণাত্মক চিহ্ন সমেত লিখলে দাঁড়াবে যথাক্রমে 00100010_2 এবং 1101111_2। এবারে সংখ্যা দুটি যোগ করা যাক।

$$\begin{array}{r} 0\,0\,1\,0\,0\,0\,1\,0_2 \\ +\,1\,1\,0\,0\,1\,1\,1\,1_2 \\ \hline 1\,1\,1\,1\,0\,0\,0\,1_2 \end{array}$$

যোগ করার পর দেখা যাচ্ছে হাতে 1 নেই এবং যোগফলটির অষ্টম বিটটিতে 1 থাকছে। কাজেই সংখ্যাটি ঋণাত্মক। সংখ্যাটি ঋণাত্মক হওয়াতে এর অষ্টম বিটটি ছাড়া প্রত্যেকটি অঙ্কের 1-এর প্রতিপূরক বের করলে তবেই সঠিক বিয়োগফলটি পাওয়া যাবে। এবং এর প্রতিপূরক নির্ণীত হলে সংখ্যাটি হবে 10001110_2 অর্থাৎ -14_{10}। অর্থাৎ মনে রাখতে হবে বিয়োগ করার পর (আসলে যে সংখ্যাটি বিয়োগ করা হবে সেই সংখ্যাটির প্রত্যেকটি অঙ্কের 1-এর প্রতিপূরক বের করে যোগ করার পর) সংখ্যাটি ঋণাত্মক হলে, আবার সংখ্যাটির অঙ্কগুলির প্রত্যেকটির 1 এর প্রতিপূরক বের করা হয়। কিন্তু সংখ্যাটি ধনাত্মক হলে আর কিছু করার প্রয়োজন হয় না। অর্থাৎ ঋণাত্মক সংখ্যা হলেই সংখ্যাটির প্রতিটি অঙ্কের 1-এর প্রতিপূরক বের করতে হয়, তা সংখ্যাটি বিয়োগ করা বা বিয়োগ ফলটি সঠিক ভাবে বের করা—যে কোনো সময়েই হোক না কেন। কাজেই যদি দুটি ঋণাত্মক সংখ্যাকে যোগ করতে হয়, সেক্ষেত্রেও সংখ্যা দুটির প্রত্যেকটি অঙ্কের 1-এর প্রতিপূরক বের করে তবেই সেই নতুন সংখ্যা দুটিকে যোগ করতে হবে। একটি উদাহরণ দিয়ে দেখানো যাক।

দুটি সংখ্যা -28_{10} এবং -34_{10} নেওয়া হল। 28_{10} এবং 34_{10} সংখ্যা দুটির সমতুল দ্বি-নিধানী সংখ্যা দুটির হবে যথাক্রমে 0011100_2 এবং 0100010_2। এবারে সংখ্যা দুটি ঋণাত্মক হওয়াতে

সংখ্যা দুটির প্রত্যেকটি অঙ্কের 1-এর প্রতিপূরক নিতে হবে। এবং এই কমপিউটারে প্রতিটি কোষ 8টি বিট যুক্ত ধরে নিয়ে এগোলে সংখ্যা দুটি হবে যথাক্রমে 11100011_2 এবং 11011101_2। এবারে ওই সংখ্যা দুটিকে অষ্টম বিট পর্যন্ত যোগ করার পরে পাওয়া যাবে—

যোগফলটির অষ্টম বিটটি 1 হওয়ায় বোঝা যাচ্ছে সংখ্যাটি ঋণাত্মক। সুতরাং আবার সংখ্যাটির অষ্টম বিটটি ছাড়া প্রত্যেকটি অঙ্কের 1-এর প্রতিপূরক বের করলে তবেই সঠিক যোগফলটি নির্ণীত হবে। এক্ষেত্রে সঠিক ফলটি হবে 10111110_2 অর্থাৎ -62_{10}।

অর্থাৎ দেখা যাচ্ছে, দুটি সংখ্যার বিয়োগ আসলে যোগেরই সমতুল। গুণ করাও বারংবার যোগ করেই নির্ণয় করা সম্ভব। আবার একটি সংখ্যাকে আর একটি সংখ্যা দিয়ে ভাগ করার প্রয়োজন হলে তা বারবার বিয়োগের সাহায্যেই করা যায় এবং বিয়োগও এক ধরনের যোগ বলেই একথা বললে ভুল হবে না যে, কমপিউটারে সব ধরনের পাটীগণিতই কেবলমাত্র যোগ করেই করা চলে এবং তিনটি লজিকাল গেট 'এণ্ড', 'অর' এবং 'নট' দিয়ে ইলেকট্রনিক সার্কিট তৈরি করেই তা করা সম্ভব।

কমপিউটারের সংকেত পদ্ধতি ঃ

কমপিউটারে সংখ্যা ছাড়াও অনেক সময় নাম, ঠিকানার মত তথ্যাদি নিয়েও কাজ করতে হয়। সেক্ষেত্রে কমপিউটারের ইংরেজি বর্ণমালা চেনার প্রয়োজন। ওই ইংরেজি বর্ণমালা ছাড়াও আরও কিছু বিশেষ চিহ্নের সঙ্গে কমপিউটারের পরিচয় ঘটানো হলে নানা ধরনের সমস্যার খুব সহজেই সমাধান করা সম্ভব। অনেক সময়ে দশমিক সংখ্যা-পদ্ধতির 0, 1, 2, 3, 4, 5, 6, 7, 8, এবং 9 এই দশটি অঙ্ককেও দ্বি-নিধানী সংখ্যা পদ্ধতিতে পরিবর্তন না করে চিহ্ন হিসেবেই রেখে দেওয়ার প্রয়োজন হয়। অর্থাৎ এই ইংরেজি বর্ণমালা, কিছু বিশেষ চিহ্ন (যেথা +, –, × প্রভৃতি) এবং দশমিকের দশটি অঙ্ক প্রত্যেককেই এক একটি আলাদা চিহ্ন হিসেবে কমপিউটারকে চেনানোর জন্য বিটের সাহায্যে বিভিন্ন সংকেত তৈরি করা হয়।

আমরা এর আগে দেখেছি 3টি বিটের সাহায্যে ৪টি পৃথক সংকেত তৈরি করা যায়। অনুরূপভাবে দেখানো যেতে পারে 4, 5 এবং 6টি বিট দিয়ে যথাক্রমে 16, 32 এবং 64টি পৃথক সংকেত তৈরি করা সম্ভব। এক একটি চিহ্নের জন্য কটি করে বিট নেওয়া হবে তা নির্ভর করবে কোন ধরনের সংকেত পদ্ধতি ব্যবহার করা হবে, তার উপরে। একটি কমপিউটার সাধারণত তিন ধরনের মধ্যে যে কোনো এক ধরনের সংকত-পদ্ধতি ব্যবহার করেঃ 6-বিট 'বিসিডি' (BCD–Binary Coded Decimal), 7-বিট 'অ্যাসকাই' (ASCII –American Standard Code for Information Interchange) এবং 8-বিট 'ইবিসিডিআইসি' (EBCDIC–Extended Binary Coded Decimal Interchange Code)। নীচের তালিকায় কয়েকটি ভিন্ন ভিন্ন চিহ্নকে এই তিন ধরনের সংকেতে দেখানো হচ্ছে।

চিহ্ন	BCD (6-বিট)	ASCII (7-বিট)	EBCDIC (8-বিট)
A	11 0001	100 0001	1100 0001
B	11 0010	100 0010	1100 0010
O	10 0110	100 1111	1101 0110
<	11 1110	011 1100	0100 1100
=	01 1101	011 1101	0111 1110
0 (zero)	00 1010	011 0000	1111 0000
1	00 0001	011 0001	1111 0001
9	00 1001	011 1001	1111 1001

এখানে লক্ষ্যণীয়, প্রত্যেকটি সংকেতে যে কোনো চিহ্নের জন্যই একই সংখ্যক বিট ব্যবহার করা হয়। যেমন, 7-বিট 'অ্যাসকাই' সংকেতে যে কোনো চিহ্নের জন্যই 7-টি বিটের দরকার। অনুরূপভাবে, 8-বিট 'ইবিসিডিআইসি' সংকত পদ্ধতিতে প্রত্যেকটি চিহ্নের জন্য 8-টি করে বিটের প্রয়োজন। একটি অক্ষর বা চিহ্ন বোঝানোর জন্যে যে কটি বিট লাগে, তাকে বাইট (Byte) বলা হয়। অর্থাৎ একটা বাইট কতকগুলি বিটের সমষ্টি। তবে সাধারণত বাইট বলতে ৪টি পরপর বিটকেই বোঝানো হয়। সুতরাং 'ইবিসিডিআইসি' সংকেত-পদ্ধতিতে একটি বাইটের সাহায্যে একটি চিহ্ন বা অক্ষর বোঝানো সম্ভব।

এখানে আর একটি কথা মনে রাখতে হবে যে, দশমিক অঙ্কগুলি যখন চিহ্ন হিসেবে থাকবে তখন একটি সংকেতে বিটের পরিস্থিতি একরকম হবে আবার সেই অঙ্কটি যখন সংখ্যা হিসেবে দেখানো হবে তখন সেই একই সংকেতে বিটের পরিস্থিতি অন্যরকম থাকবে। 35 সংখ্যাটির কথা ধরা যাক। এই দশমিক সংখ্যাটিকে কমপিউটারে

দুভাবে রাখা চলে। সংখ্যাটির সাহায্যে কোনো পাটীগণিত করবার সময়ে সংখ্যাটিকে দ্বি-নিধানী সংখ্যায় পরিবর্তন করে রাখা হয়। আবার 35-কে 3 এবং 5 দুটি আলাদা চিহ্ন হিসেবেও রাখা সম্ভব। একটি 16 বিটের কমপিউটারে ওই দুভাবে রাখলে কিরকম দেখাবে তা এবারে দেখানো হচ্ছে। 35-কে দ্বি-নিধানী সংখ্যায় লিখলে দাঁড়ায়

		ভাগশেষ
2	35	
2	17	1
2	8	1
2	4	0
2	2	0
	1	0

100011_2। এই ধনাত্মক সংখ্যাটিকে 16 বিটের একটি কোষে রাখলে বিটগুলির অবস্থা হবে 0000000000100011। এবারে 3 এবং 5 ওই দুটি চিহ্ন 'ইবিসিডিআইসি' সংকেতে লিখলে হবে যথাক্রমে 1111 0011 এবং 1111 0101। এই দুটি চিহ্নকে একই সঙ্গে একটি কোষে রাখা সম্ভব। এবারে এই দুটি একটি কোষে রাখলে কোষটির বিটগুলি হবে 1111 0011 1111 0101। কাজেই 35-কে সংখ্যা হিসেবে মনে করলে বিটগুলির পরিস্থিতি একরকম থাকবে। আবার 3 ও 5 আলাদা চিহ্ন হিসেবে রাখলে বিটগুলির পরিস্থিতি হবে অন্যরকম। কোনো ব্যক্তির ঠিকানার অঙ্কগুলি কমপিউটারে রাখবার সময়েই 3 এবং 5 বা অন্যান্য সংখ্যাকে আলাদা চিহ্ন হিসেবে রাখার প্রয়োজন হয়। এক্ষেত্রে এই অঙ্কগুলি দিয়ে কোনো পাটীগণিত করার প্রয়োজন হবে না, এগুলি কেবল স্মৃতিতে রেখে দিয়ে পরে কমপিউটারের নির্গম অংশ দিয়ে ছাপাতে হবে।

সংকেত-পদ্ধতি সম্বন্ধে আলোচনা শেষ করার আগে এই প্রসঙ্গে বলা যেতে পারে, টেলিগ্রাফে যেমন ডট এবং ড্যাশ-এর সাহায্যে চিহ্নের সাংকেতিক রূপ দেওয়া হয়, কমপিউটারেও তেমনি 0 এবং 1-এর সাহায্যে তা করা হয়ে থাকে।

কমপিউটারের ভাষা
ও
সফ্‌টওয়ার

কমপিউটারের তিন স্তরের ভাষা ঃ

একজন মানুষ আর একজন মানুষের সঙ্গে যোগাযোগ রক্ষা করে ভাষার মাধ্যমে। কেউ যোগাযোগ রক্ষা করে বাংলায়, কেউ ইংরেজিতে, কেউ বা আবার অন্য কোনো ভাষায়। কমপিউটারের সঙ্গে কাউকে যোগাযোগ রক্ষা করতে হলে কমপিউটারের বোধগম্য কোনো ভাষার ব্যবহার প্রয়োজন। এখন যে সব কমপিউটার পাওয়া যায়, তাতে কেবলমাত্র 0 এবং 1 বোঝাবার ব্যবস্থা আছে। সুতরাং কমপিউটারের ভাষাও এই 0 এবং 1 দিয়েই তৈরি। ওই 0 এবং 1 দিয়ে তৈরি ভাষাকে যন্ত্রের ভাষা (Machine Language) বলা হয়।

কমপিউটারের সাহায্যে কোনো সমস্যা সমাধান করার জন্য যখন কমপিউটারের সঙ্গে যোগাযোগ রক্ষা করার প্রয়োজন ঘটে, তখন একটি কর্মসূচী তৈরি করা হয়। এই কর্মসূচীকেই 'প্রোগ্রাম' (Program) বলে। এই প্রোগ্রাম কয়েকটি নির্দেশের সমষ্টি। এই নির্দেশগুলি 0 এবং 1-এর সংমিশ্রণে সংকেতাবদ্ধ। একটি নির্দেশ সাধারণত দুটি অংশে বিভক্ত। এর মধ্যে প্রথম অংশে যোগ বা বিয়োগ করা বা কোনো সংখ্যা পড়ার মত যে সব কাজ করতে হবে, সেই ধরনের নির্দেশ উল্লেখ করা থাকে। আর প্রথম অংশ যে কাজটি করতে বলা হচ্ছে, সেই কাজটির জন্য যদি একটি স্মৃতিকোষ থেকে কোনো তথ্য (data) নিয়ে আসতে হয় তবে দ্বিতীয় অংশে থাকে সেই স্মৃতিকোষের ঠিকানা। একটি কাজের ধরন বোঝানোর জন্য একটি নির্দেশে কিছু সংখ্যক বিট নির্দিষ্ট রাখা হয়। কতগুলি বিট নির্দিষ্ট থাকবে তার সংখ্যা এক এক ধরনের কমপিউটারে এক একরকম। যদি কোনো কমপিউটারে 5টি বিট নির্দিষ্ট থাকে, তবে কাজের ধরনকে বোঝাবার জন্য সেই

কমপিউটারে 2_5 অর্থাৎ 32টি বিভিন্ন ধরনের কাজ করা সম্ভব। একটি 32 বিট কমপিউটারে বিভিন্ন ধরনের কাজ বোঝাতে সাধারণত 8টি বিট নির্দিষ্ট রাখা হয়। ওই কমপিউটারে 2_8 অর্থাৎ 256টি বিভিন্ন কাজের ধরন থাকা সম্ভব। কাজেই ওই কমপিউটারে কোনো সমস্যা সমাধান করতে হলে যে প্রোগ্রাম তৈরি করা হবে সেখানে 256টি বিভিন্ন কাজের ধরনের বাইরে আর কোনো নতুন ধরনের কাজ নিয়ে আসা যায় না। কিন্তু এর অর্থ এই নয় যে, প্রোগ্রামে সবচেয়ে বেশি 256টি নির্দেশই থাকবে। কোনো একটি প্রোগ্রামে কয়েক হাজার নির্দেশও রাখা যায়, যদিও বিভিন্ন ধরনের নির্দেশ কখনোই 256 টির বেশি রাখা চলে না। কিন্তু একই ধরনের নির্দেশ বহুবার থাকতে পারে, যেমন, যোগ বা বিয়োগ করার মত নির্দেশটি একটি প্রোগ্রামে অনেকবারই ব্যবহার করা সম্ভব।

একটি কমপিউটারে বিভিন্ন ধরনের কাজ বোঝানোর জন্য কতগুলি বিট নির্দিষ্ট থাকবে তা সম্পূর্ণ নির্ভর করে সেই কমপিউটারের একটি স্মৃতিকোষে বিটের সংখ্যা এবং সেই সঙ্গে ওই কমপিউটারের গঠনের উপরে। একটি নির্দেশের জন্য কোনো কোনো কমপিউটারে 1 থেকে 3টি পর্যন্ত কোষের প্রয়োজন হতে পারে। নীচের চিত্রে একটি কোষে একটি নির্দেশ কি ভাবে থাকে তা দেখানো হচ্ছে।

কাজের ধরন বোঝানোর জন্য নির্দেশ	একটি স্মৃতিকোষের ঠিকানা যেখানে ওই কাজটি করার জন্য যে তথ্যের (data) দরকার তা সঞ্চয় করা আছে।

এখানে মনে করা হচ্ছে, এই কমপিউটারে একটি নির্দেশ রাখতে 16 বিটের একটি স্মৃতিকোষই যথেষ্ট এবং কাজের ধরন বোঝানোর জন্যে 5টি বিট নির্দিষ্ট আছে। এই কমপিউটারে বিভিন্ন ধরনের কাজ এবং সেইসব কাজ বোঝানোর জন্য 0 এবং 1-এর সংকেতাবদ্ধ রূপের কিছু নমুনা নীচে দেওয়া হল।

কাজের ধরন	0 এবং 1 এর সংকেতাবদ্ধরূপ
1. কোন সংখ্যা বা নাম অনুপ্রবেশ অংশের মাধ্যমে স্মৃতিকোষে আনার জন্য।	10000
2. নিয়ন্ত্রণ অংশের বা সিপিউর রেজিস্টারে কিছু তথ্য স্মৃতিকোষ থেকে নিয়ে এসে রাখার জন্য।	10001

3.	রেজিস্টারের সংখ্যাটির সংঙ্গে কোনো একটি স্মৃতিকোষে সঞ্চিত সংখ্যার যোগ করার জন্য। এক্ষেত্রে যোগফলটি রেজিস্টারেই থাকবে ধরে নেওয়া হচ্ছে।	10010
4.	রেজিস্টারের সঞ্চিত কোনো সংখ্যা কোনো একটি স্মৃতিকোষে আনার জন্যে।	10011
5.	কোনো একটি স্মৃতিকোষের সঞ্চিত কোনো সংখ্যা বা নাম নির্গম অংশের মাধ্যমে ছাপানোর জন্য।	10100
6.	সব কাজ হয়ে যাওয়ার পরে থামার জন্য।	11111

এখন একটি সমস্যা সমাধানের জন্য ওই যন্ত্রের ভাষায় কি ধরনের নির্দেশ দিতে হবে তা দেখানো যেতে পারে। দুটি সংখ্যার যোগফল নির্ণয় করাই এখানে সমস্যা হিসেবে মনে করা যাক। দুটি সংখ্যাকে যোগ করার আগে সে দুটিকে এক এক করে অনুপ্রবেশ অংশের মাধ্যমে দুটি স্মৃতিকোষে সঞ্চয় করা হয়। এরপর একটি সংখ্যাকে সিপিউ-এর একটি রেজিস্টারে এনে দ্বিতীয় সংখ্যাটির সঙ্গে যোগ করতে হবে। যোগফলটি রেজিস্টারেই থাকবে। কাজেই রেজিস্টার থেকে যোগফলটি কোনো একটি স্মৃতিকোষে সঞ্চয় করে রাখা হয়। এরপর ওই স্মৃতিকোষের সংখ্যাটি নির্গম অংশের মাধ্যমে ছাপানোর পর থামার জন্য নির্দেশ দিতে হবে। উপরের সমস্যাটি সমাধানের জন্য এখন যন্ত্রের ভাষায় নির্দেশগুলি পরপর লিখে দেখানে হচ্ছে।

1.	10000	00000100000
2.	10000	00000100001
3.	10001	00000100000
4.	10010	00000100001
5.	10011	00000100010
6.	10100	00000100010
7.	11111	00000000000

এখানে ধরে নেওয়া হচ্ছে ওই কমপিউটারে প্রত্যেকটি কোষে 16টি করে বিট আছে, প্রত্যেকটি নির্দেশে কাজের ধরন বোঝানোর জন্য 16 সংখ্যক থেকে 12 সংখ্যক বিট ব্যবহার করা হয়েছে এবং 1 থেকে 11 সংখ্যক বিটের সাহায্যে একটি স্মৃতিকোষের ঠিকানা নির্দিষ্ট রয়েছে। প্রথম নির্দেশটিতে 16 সংখ্যক থেকে 12 সংখ্যক বিটে 10000 থাকায় একটি সংখ্যা অনুপ্রবেশ অংশের মাধ্যমে পড়ে একটি স্মৃতিকোষে রাখাই ওই নির্দেশটির কাজের ধরন হবে। এই নির্দেশের 1 থেকে 11 সংখ্যক বিটে 00000100000_2 থাকায় স্মৃতিকোষটির

ঠিকানাটি 32_{10} । যে সংখ্যাটি পড়া হবে সেটি ধনাত্মক বা ঋণাত্মক কোনো একটি সংখ্যা হতে পারে । মনে করা যাক, সংখ্যাটি 21_{10} । এই সংখ্যাটির সমতুল দ্বি-নিধানী সংখ্যাটি হল—

		ভাগশেষ
2	21	
2	10	1
2	5	C
2	2	1
	1	0

10101_2 । অর্থাৎ 32 সংখ্যক কোষের বিটগুলি হবে 00000 00000010101 । 16 সংখ্যক বিটটিতে 0 থাকায় বোঝা যায়, সংখ্যাটি ধনাত্মক । দ্বিতীয় নির্দেশটিতেও প্রথম নির্দেশটির মত একটি সংখ্যা পড়া হচ্ছে । এক্ষেত্রে মনে করা যাক সংখ্যাটি 30_{10} । কাজেই এই সংখ্যাটির সমতুল দ্বি-নিধানী সংখ্যাটি 11110_2 হওয়াতে 33 সংখ্যক কোষে (দ্বিতীয় নির্দেশের কোষের ঠিকানাটি 33) 0000000000011110 থাকবে । তৃতীয় নির্দেশটিতে 32 সংখ্যক কোষের সংখ্যাটিকে সিপিউ অংশের রেজিস্টারে আনার জন্য বলা হয়েছে । কাজেই 21_{10} সংখ্যাটির সমতুল দ্বি-নিধানী সংখ্যাটি রেজিস্টারে চলে আসবে । চতুর্থ নির্দেশটি অনুসারে 33 সংখ্যক কোষের সংখ্যাটির সঙ্গে রেজিস্টারের সংখ্যাটি যোগ করা হবে এবং যোগফলটি রেজিস্টারেই থাকবে । এরপর পঞ্চম নির্দেশটি অনুসারে যোগফলটি রেজিস্টার থেকে 34 সংখ্যক স্মৃতিকোষে রাখা হবে । 6 সংখ্যক নির্দেশটিতে 34 সংখ্যক স্মৃতিকোষের সঞ্চিত সংখ্যাটিকে ছাপানোর কথা বলা আছে এবং সর্বশেষ নির্দেশটিতে থামতে বলা হচ্ছে । থামানোর নির্দেশটি না থাকলে এর পরের স্মৃতিকোষে যা থাকবে নিয়ন্ত্রণ অংশটি তা নিয়ে এসে সেখানকার নির্দেশ অনুসারে কাজ করার চেষ্টা করবে । এই সব স্মৃতিকোষে পূর্বের কোনো সমস্যার সমাধানের জন্য কিছু নির্দেশ থাকতেও পারে । কাজেই সেইসব নির্দেশ অনুসারে কমপিউটার কাজ করে যেতে থাকবে । এর ফলে প্রথম 6টি নির্দেশ অনুসারে এই সমস্যাটির সঠিক সমাধান করার পর কিছু এলোমেলো কাজ চলতে পারে । এইজন্য কোনো সমস্যা সমাধানের নির্দেশগুলি দেওয়ার পরে কমপিউটারকে থামানোর জন্যও নির্দেশ দেওয়ার প্রয়োজন আছে ।

উপরের সমস্যাটির সমাধানের জন্য মোট 7টি নির্দেশের অর্থাৎ 7টি কোষের প্রয়োজন । এ ছাড়াও সংখ্যাগুলি রাখার জন্য আরও তিনটি স্মৃতিকোষের প্রয়োজন । কাজেই এই সমস্যাটির জন্য মোট

10টি স্মৃতিকোষের দরকার। কমপিউটারের স্মৃতিকোষে সবকটি নির্দেশ সঞ্চয় করার পরে নিয়ন্ত্রণ অংশটি এক এক করে নির্দেশগুলি নিয়ে এসে কাজ করতে থাকে এবং কাজের ধরনের উপর নির্ভর করে কোনো একটি অংশকে কাজটি করে দেবার জন্য নির্দেশ পাঠায়। একটি কাজ হয়ে যাওয়ার পর নিয়ন্ত্রণ অংশটি আবার পরের নির্দেশটি নিয়ে এসে তা কি ধরনের কাজ বিশ্লেষণ করে দেখে।

প্রথম দিকের কমপিউটারে এই যন্ত্রের ভাষাতেই কর্মসূচী রচনা করা হত। কমপিউটার যন্ত্রে যেমন এক একটি জেনারেশনের আবির্ভাব ঘটেছে তেমনি কর্মসূচী রচনার ক্ষেত্রেও কমপিউটারের ব্যবহৃত ভাষারও ক্রমশ উন্নতি হয়েছে। যন্ত্রের ভাষা বা মেশিন-ভাষা ব্যবহার যথেষ্ট শ্রমসাধ্য এবং সময়সাপেক্ষ। এর সাহায্যে প্রোগ্রাম রচনা করা বেশ জটিল ব্যাপার। এই ভাষাতে প্রত্যেকটি বিভিন্ন ধরনের নির্দেশের 0 এবং 1-এর সংকেতাবদ্ধ রূপ মনে রাখতে হয়। এর সঙ্গে ইংরেজি বা ওরকম অন্য কোনো ভাষার মিল না থাকাতে মনে রাখার কাজটা সহজ নয়। কাজেই এরপরে আর এক ধরনের ভাষা প্রবর্তিত হল। এই ভাষা যন্ত্রের ভাষারই অন্য এক রূপ। এক্ষেত্রে 0 এবং 1-এর পরিবর্তে স্মৃতি-সহায়ক (Mnemonic) নাম ব্যবহার করা হয়। ওই ভাষা প্রতীকী ভাষা (Assembly Language) নামে অভিহিত। এই ভাষার প্রধান সুবিধে, যন্ত্রের ভাষায় কোনো একটি নির্দেশে যোগ করতে বলা হলে যোগের জন্য যেমন 0 এবং 1-এর সংকেতাবদ্ধ রূপটি মনে রাখতে হয়, এখানে তার কোনো প্রয়োজন হয় না। এক্ষেত্রে নির্দেশে স্মৃতি সহায়ক নাম AD লেখা যেতে পারে (ADD এই শব্দটির প্রথম দুটি অক্ষর নিয়ে AD বলা যায়) এবং ওই AD শব্দটি মনে রাখা অনেক সহজ। এরপর যন্ত্রের ভাষায় যে সমস্যাটির সমাধান করা হয়েছে সেটিই ওই ধরনের একটি প্রতীকী ভাষাতে লিখে দেখানো যেতে পারে।

1. RE A (এখানে READ এর প্রথম দুটি অক্ষর নিয়ে RE লেখা হয়েছে এবং A একটি স্মৃতিকোষের ঠিকানা হিসেবে ধরা হয়েছে)
2. RE B (এখানে আর একটি সংখ্যা পড়ে B নামের কোষে রাখতে বলা হচ্ছে)
3. LD A (LOAD শব্দটির দুটি অক্ষর নেওয়া হয়েছে। এটির সাহায্যে বোঝানো হয়, A স্মৃতিকোষের সঞ্চিত সংখ্যাটি রেজিস্টারে রাখা হল)
4. AD B (ADD-এর দুটি অক্ষর নেওয়া হচ্ছে। রেজিস্টারে রাখা সংখ্যাটির সঙ্গে B-এর সঞ্চিত সংখ্যাটি যোগ করতে বলা হল)

5. SR C (STORE এর দুটি অক্ষর দিয়ে বোঝানো হচ্ছে রেজিস্টারের যোগফলটি C নামের স্মৃতিকোষে রাখতে হবে। এখানে ST না লিখে SR নেওয়ার কারণ ST বলতে STOP বোঝানো হবে)

6. PR C (PRINT এর প্রথম দুটি অক্ষর নেওয়া হল)

7. ST (STOP এর প্রথম দুটি অক্ষর নেওয়া হল)

এখানে মূল নির্দেশ ব্যক্ত শব্দটির দুটি অক্ষরের সাহায্যে কাজের ধরন বোঝানো হচ্ছে। পরে যে অক্ষরটিকে নেওয়া হয়েছে, সেটি স্মৃতিকোষের ঠিকানার নাম হিসেবে ব্যবহৃত। এবার এক একটি নির্দেশ লক্ষ্য করা যাক। প্রথম নির্দেশটিতে অনুপ্রবেশ অংশের মাধ্যমে একটি সংখ্যা পড়ে তা একটি স্মৃতিকোষে রাখার কথা বলা হচ্ছে। এই স্মৃতিকোষটিকে প্রোগ্রামে A নামে ডাকা যেতে পারে। পরের নির্দেশটিতে অন্য একটি সংখ্যা পড়ে আর একটি স্মৃতিকোষে রাখা হবে। এর নামে দেওয়া যাক B। এরপর A নামের স্মৃতিকোষের সংখ্যাটি রেজিস্টারে আনতে হবে। এখানে উল্লেখ করা দরকার, রেজিস্টারে কোনো তথ্য স্মৃতিকোষ থেকে এনে রাখা যায় আবার প্রয়োজনমতো কোনো তথ্য রেজিস্টার থেকে স্মৃতিকোষেও নিয়ে যাওয়া চলে। সাধারণত অনুপ্রবেশ অংশের সাহায্যে কিছু পড়ে রেজিস্টারে তা সরাসরি রাখা যায় না, আবার রেজিস্টার থেকে সরাসরি নির্গম অংশের মাধ্যমে কিছু ছাপানোও সম্ভব নয়। চতুর্থ নির্দেশে রেজিস্টারের সংখ্যাটির সঙ্গে B নামের স্মৃতিকোষের সংখ্যাটির যোগ দিয়ে যোগফলটি স্মৃতিকোষেই রাখা হবে। পরবর্তী নির্দেশ অনুসারে যোগফলটিকে C নামের স্মৃতিকোষে রেখে 6 সংখ্যক নির্দেশ অনুসারে C-এর সংখ্যাটি ছাপানো হবে এবং 7 সংখ্যক নির্দেশে এসে থামতে বলা হবে। এই ভাষায় প্রোগ্রাম লেখার ক্ষেত্রে কিছুটা সুবিধে হলেও প্রোগ্রামটি লিখতে প্রায় যন্ত্রের ভাষায় মতই সময় লাগে। এই ভাষায় লেখা প্রোগ্রাম কিন্তু কমপিউটার সরাসরি বুঝতে পারে না। এর কারণ কমপিউটারের নিয়ন্ত্রণ অংশটি AD বলে কিছু বোঝে না, AD-এর জন্য 0 এবং 1-এর সংকেতাবদ্ধ রূপটিই কমপিউটারের সার্কিটে চেনানো আছে। কাজেই এই ভাষায় লেখা কোনো প্রোগ্রাম কমপিউটারকে বুঝতে হলে আর একটি প্রোগ্রামের প্রয়োজন। এই প্রোগ্রামটি অনুবাদকের (Translator) কাজ করে। প্রতীকী ভাষায় লেখা প্রত্যেকটি নির্দেশ এই অনুবাদক কমপিউটার যন্ত্রটির ভাষায় অনুবাদ করে দেয়। অনুবাদের পরে সবকটি নির্দেশই সেই কমপিউটারের যন্ত্রের ভাষায় রূপান্তর লাভ করে এবং এরপরই কমপিউটার কাজ করতে পারে। এই অনুবাদক প্রোগ্রামটিকে প্রতীকী ভাষান্তরক বা অ্যাসেমব্লার

(Assembler) বলে। যন্ত্রের ভাষার মতই প্রতীকী ভাষান্তরকও কমপিউটার নির্ভর। এক এক ধরনের কমপিউটারের জন্য এক একরকমের প্রতীকী ভাষান্তরক থাকে। যে কোম্পানি কমপিউটারটি বিক্রি করে তারাই কমপিউটারের সঙ্গে এই প্রোগ্রামটিও ক্রেতাকে দিয়ে দেয়। কমপিউটারে প্রোগ্রামটি অনেক ভাবেই রাখা যেতে পারে। স্মৃতির আলোচনাতে 'রম' (ROM) স্মৃতি সম্বন্ধে বলা হয়েছিল। ওই ভাষান্তরকটি 'রম' স্মৃতিতে রাখা চলে। এর ফলে প্রোগ্রামটি কখনোই স্মৃতি থেকে মুছে যাবে না। আবার কমপি-উটারের সঙ্গে ডিস্ক কিংবা টেপ থাকলে তাতেও এটি সঞ্চিত রাখা সম্ভব। এরপর প্রয়োজনমত এটিকে কমপিউটারের স্মৃতিতে নিয়ে এসে কাজ করা যায়।

বর্তমানে কমপিউটারের অগ্রগতির সঙ্গে এর ব্যবহারও প্রচুর পরিমাণে বেড়ে গেছে। বৈজ্ঞানিকেরা এবং ইঞ্জিনিয়ারেরা তাঁদের প্রয়োজনে কমপিউটারের ব্যবহার শুরু করলেন। ব্যাপক প্রচলনের ক্ষেত্রে কমপিউটার কোম্পানিগুলি একটা বিষয় অনুভব করতে আরম্ভ করে যে, এই সব যন্ত্রের ভাষা এবং প্রতীকী ভাষায় প্রোগ্রাম লেখা এঁদের পক্ষে রীতিমতো অসুবিধেজনক। এবং শেষ পর্যন্ত কমপিউটার বিক্রী ব্যাহত হতে থাকে। কমপিউটারে কাজ করার জন্য আরও সহজ ও সরল ভাষার প্রয়োজন হয়ে পড়ে। যন্ত্রের ভাষা এবং প্রতীকী ভাষা কমপিউটার নির্ভর। কিন্তু এখন এমন ভাষার প্রয়োজন যা কমপিউটার নির্ভর নয়। অর্থাৎ ওই ভাষাতে কোনো সমস্যা সমাধানের জন্য প্রোগ্রাম লিখলে তা যে কোনো কমপি-উটারেই ব্যবহার করা সম্ভব। এই ধরনের ভাষাকে উচ্চ পর্যায়ের ভাষা (High level language) বা প্রক্রিয়া ভাষা (Procedural language) বলে অভিহিত করা হয়েছে। ওই ভাষা এমন হওয়া উচিত যার ফলে এর সাহায্যে অনেক সহজে এবং অতি দ্রুত প্রোগ্রাম লেখা সম্ভব। শেষ পর্যন্ত এই ধরনের ভাষা তৈরির জন্য একটি কমিটি গঠন করা হল। সেই কমিটি 1960 সালে 'অ্যালগল' (ALGOL–Algorithmic language) ভাষা তৈরি করে। কিন্তু ওই ভাষা বেরোনোর আগেই 1958 সালে IBM কোম্পানি তাদের কমপিউটার বিক্রি বাড়ানোর তাগিদে 'ফরট্রান' (FORTRAN –Formula Translation) নামে একটি ভাষার প্রচলন করে এবং বেশি জায়গাতে IBM-এর কমপিউটার থাকাতে এই ভাষারই বহুল প্রচার ঘটে। আমাদের দেশেও 1963 সালের শেষের দিক থেকে এই ভাষা শেখানোর ব্যবস্থা হয়। এবারে যন্ত্রের ভাষাতে যে সমস্যাটির সমাধান দেখানো হয়েছে সেই সমস্যাটির সমাধান 'ফরট্রানে' লিখলে কিরকম হবে তা নীচে দেখানো হচ্ছে।

```
1. READ A, B
```

```
2. C = A + B
3. PRINT C
4. STOP
```

এক্ষেত্রে দুটি সংখ্যা একটি নির্দেশের সাহায্যেই পড়ে দুটি আলাদা স্মৃতিকোষে রাখা হচ্ছে। যোগ করার জন্য এই ভাষাতে যোগ চিহ্ন (+) ব্যবহার করেই করা যায় এবং একটি নির্দেশেই যোগ করা এবং যোগফলটি একটি স্মৃতিকোষে রাখার ব্যবস্থা আছে। এরপর সংখ্যাটি ছাপিয়ে থামতে বলা হয়েছে। আগে যেখানে যন্ত্রের ভাষা এবং প্রতীকী ভাষাতে সমস্যাটি সমাধানের জন্য 7টি নির্দেশ প্রয়োজন হয়েছে, এই ভাষাতে 4টি নির্দেশেই তা করা সম্ভব। কিন্তু সংখ্যাগুলি রাখার জন্যও সব ভাষাতেই একই সংখ্যক স্মৃতিকোষের প্রয়োজন।

এই দুই উচ্চ পর্যায়ের ভাষায় বৈজ্ঞানিক এবং ইঞ্জিনিয়ারদের প্রোগ্রাম লেখা খুব সহজ হয়ে যায়। এতে একটি সমস্যা সমাধানের জন্য আগের ভাষার চেয়ে অনেক কম সংখ্যক নির্দেশের প্রয়োজন। এর কারণ একটি নির্দেশেই এখানে অনেক কাজ করা চলে। বড় বড় সূত্র এই ভাষাতে একটি নির্দেশেই লেখা যেতে পারে। 'ফরট্রান' নামটি ওই কারণেই ব্যবহার করা হয়েছে। ওই দুই ভাষাতেই ইংরেজি শব্দের ব্যবহার থাকলেও পুরোপুরি ইংরেজি ভাষার সঙ্গে এর যোগ নেই। এরপর ব্যবসা-বাণিজ্য সংক্রান্ত ব্যাপারে এবং সকলের কাছে সহজবোধ্য করে তোলার জন্যে একটি ভাষার প্রয়োজন হয়। এই কারণে 1966 সালে 'কোবল' (COBOL –Common Business Oriented Language) ভাষা আসে। ওই ভাষার সঙ্গে ইংরেজি ভাষার অনেকটাই মিল রয়েছে। ওই ভাষাতে লেখা যে কোনো প্রোগ্রাম পড়ে সহজেই বোঝা সম্ভব। এইসব ভাষা ছাড়া আরও অনেক উচ্চ পর্যায়ের ভাষার চল আছে। 'পাসকাল' (Pascal), 'পি এল/ 1' (PL/1), 'সি' (C) প্রভৃতি। আজকাল যে সকল মিনি, মাইক্রো বা পার্সোনাল কমপিউটার বেরিয়েছে তার জন্য 'বেসিক' (BASIC–Begineers All Purpose Symbolic Ints-truction Code) ভাষার চল হয়েছে। 'ফরট্রানে'র সঙ্গে ওই ভাষাটির কিছুটা মিল লক্ষ্য করা যায়। যাঁরা নিজেদের কোনো সমস্যার সমাধানের জন্য কমপিউটার ব্যবহার করবেন তাঁদের পক্ষে ওই প্রক্রিয়া ভাষা শেখা যাতে খুব বেশি কষ্টকর না হয় সেকথা মনে রেখে এই 'বেসিক' ভাষাটি তৈরি করা হয়েছে। ওই ভাষাটি সম্বন্ধে বিস্তারিত আলোচনা পরে করা হবে।

এইসব প্রক্রিয়া ভাষায় লেখা প্রোগ্রামকে 'উৎস' (source) প্রোগ্রাম বলে। কিন্তু কমপিউটার সরাসরি ওই সব ভাষা বুঝতে

পারে না। ওইসব ভাষায় লেখা নির্দেশগুলি কমপিউটার বোধ্য ভাষায় রূপান্তর করার জন্য 'সংকলক' বা কমপাইলার (Compiler) নামে একটি প্রোগ্রামের প্রয়োজন। এর কাজও অনেকটা প্রতীকী ভাষান্তরকের মত। 'বেসিক' ভাষায় লেখা নির্দেশগুলি যে কমপিউটারে চালানো হবে, বেসিক কমপাইলার সেই ভাষাকে কমপিউটারের ভাষায় অনুবাদ করে দেবে। বিভিন্ন কমপিউটারের জন্য 'বেসিক' কমপাইলারটি হবে ভিন্ন ভিন্ন। যদিও বিভিন্ন কমপাইলার সকলেই বেসিক ভাষায় লেখা নির্দেশগুলি পড়ছে কিন্তু এক এক ধরনের কমপিউটারের ভাষা এক এক রকমের এবং একটি কমপাইলার কেবলমাত্র একটি ভাষাতেই অনুবাদ করতে পারে বলে বিভিন্ন কমপিউটারের জন্য ভিন্ন ভিন্ন কমপাইলারের প্রয়োজন। 'বেসিক' ভাষায় কম সংখ্যক নির্দেশ থাকায় ওই কমপাইলারগুলি অন্যান্য ভাষার কমপাইলারের চেয়ে ছোট হয়ে থাকে। এই কারণে ছোট ছোট কমপিউটারে এর প্রচলন বেশি। অবশ্য ছোট কমপিউটারে অন্যান্য ভাষার কমপাইলারও থাকে। এখানে একটা কথা উল্লেখ করা দরকার, যে কোম্পানি কমপিউটারটি বিক্রি করছে প্রতীকী ভাষান্তরকের মত সংকলকগুলির জোগানও তারাই করে থাকে। একটি ভাষার কমপাইলার প্রোগ্রামটি যেমন এক এক ধরনের কমপিউটারের জন্য এক এক রকম, তেমনি বিভিন্ন ভাষার কমপাইলার প্রোগ্রামগুলি আবার একই কমপিউটারে ভিন্ন ভিন্ন। প্রতীকী ভাষান্তরকের মত কমপাইলারগুলিও 'রম' স্মৃতি কিংবা ডিস্ক ও টেপে থাকতে পারে। এইসব কমপাইলার প্রোগ্রাম সাধারণত কমপিউটার বিশেষজ্ঞর দ্বারা লেখা হয়।

অনেক পার্সোনাল কমপিউটারে বেসিক ভাষার কমপাইলার (Compiler)-এর পরিবর্তে ওই ভাষার ইন্টারপ্রিটার (Interpreter) থাকে। কমপাইলার এবং ইন্টারপ্রিটারের মধ্যে পার্থক্য আছে। কমপাইলার প্রথমে একটি বেসিক প্রোগ্রামের সব কটি নির্দেশে কোনো ব্যাকরণগত ভুল আছে কিনা দেখে নেয়। যদি কোনো ধরনের ভুল না থাকে তাহলে নির্দেশগুলিকে কমপাইলার ওই কমপিউটারের যন্ত্রের ভাষায় অনুবাদ করে এবং তারপর প্রয়োজন-মত যন্ত্রের ভাষায় পরিবর্তিত নির্দেশ অনুসারে কাজ করে।

কিন্তু ইন্টারপ্রিটার বেসিক প্রোগ্রামটির এক একটি করে নির্দেশ নিয়ে এগোয়। প্রথমে তা একটি নির্দেশ নিয়ে সেই নির্দেশে কোনো ব্যাকরণগত ভুল আছে কিনা পরীক্ষা করে এবং ভুল না থাকলে নির্দেশটিকে কমপিউটারটির যন্ত্রের ভাষায় অনুবাদ করে পরিবর্তিত নির্দেশ অনুসারে কাজ করে। এরপর প্রোগ্রামের ওই নির্দেশটির পরের নির্দেশ নিয়ে সে প্রথমবারের মত একইভাবে কাজ করে চলে। এইভাবে প্রোগ্রামের পরপর নির্দেশগুলির বেলায় কাজ চলতে থাকে।

কমপাইলারের ক্ষেত্রে সুবিধা এই যে, একবারই সবকটি নির্দেশ যন্ত্রের ভাষায় অনুবাদ করার পরে কমপিউটার কাজ শুরু করে। ইন্টারপ্রিটারের বেলায় তা নয়। এক্ষেত্রে কোনো একটি বেসিক প্রোগ্রামে কয়েকটি নির্দেশ যদি বারবার করার প্রয়োজন হয়, তাহলে যে কবার নির্দেশগুলি করার কথা, সে কবারই ওই প্রত্যেকটি নির্দেশ অনুবাদ করতে হবে। অথচ কমপাইলারের ক্ষেত্রে একবার নির্দেশগুলি অনুবাদ করলেই কাজ হয়ে যায়।

কমপিউটারের বিভিন্ন ধরনের সফ্‌ট্‌ওয়ার ঃ

'হার্ডওয়ার' (Hardware) এবং 'সফ্‌ট্‌ওয়ার' (Software) ওই দুটি শব্দ প্রায়ই কমপিউটারের আলোচনাতে হয়। ইলেকট্রিকাল, মেকানিকাল এবং ইলেকট্রনিক অংশের সাহায্যে যে কমপিউটার যন্ত্র তৈরি হয় সেই যন্ত্রটিকেই হার্ডওয়ার হিসেবে চিহ্নিত করা হয়ে থাকে। কমপিউটার হার্ডওয়ারটির সাহায্যে কেবলমাত্র কিছু মৌলিক কাজ করা সম্ভব এবং সে কাজও যন্ত্রের ভাষা অর্থাৎ 0 এবং 1 দ্বারা করা চলে। সফ্‌ট্‌ওয়ার হল কতগুলি প্রোগ্রাম। এদের সাহায্যে কমপিউটার হার্ডওয়ারটিকে খুব সহজেই ব্যবহার করা যায়। হার্ডওয়ারটি না হলে যেমন কমপিউটার ব্যবহার করাই চলে না, সফ্‌ট্‌ওয়ার না থাকলেও তেমনি কমপিউটার ব্যবহার বেশ কষ্টসাধ্য হয়ে পড়ে। বলা যেতে পারে, হার্ডওয়ার এবং সফ্‌ট্‌ওয়ার একে অন্যের পরিপূরক। বর্তমানে সফ্‌ট্‌ওয়ারের দৌলতেই কমপিউটারের ব্যবহার প্রচুর পরিমাণে বেড়ে গেছে। কিন্তু হার্ডওয়ারটিকে যেমন স্পর্শ করা যায়, সফ্‌ট্‌ওয়ারকে তেমনি স্পর্শ করা যায় না। এর প্রয়োজনীয়তা কেবলমাত্র অনুভবের বিষয়। হার্ডওয়ার অর্থাৎ কমপিউটার যন্ত্র সম্বন্ধে আমরা ইতিপূর্বে বিস্তারিত আলোচনা করেছি। এবারে সফ্‌ট্‌ওয়ার সম্বন্ধে কিছু আলোচনা করা যাক। সফ্‌ট্‌ওয়ার সাধারণত দু ধরনের, যেমন 'সিসটেম সফ্‌ট্‌ওয়ার' (System Software) এবং 'অ্যাপলিকেশন সফ্‌ট্‌ওয়ার' (Application Software)।

যে সকল প্রোগ্রাম কিছু বিশেষ ধরনের সমস্যার সমাধান করে সে সব প্রোগ্রামকে 'আপলিকেশন সফ্‌ট্‌ওয়ার' বলে। দৃষ্টান্তস্বরূপ, মাধ্যমিক অথবা উচ্চ-মাধ্যমিক পরিক্ষার্থিদের নম্বর অনুসারে নাম সাজানোর কথা বলা যেতে পারে। আবার যে সব প্রোগ্রামের সাহায্যে কমপিউটারকে সহজে ব্যবহার করা চলে তাদের 'সিসটেম সফ্‌ট্‌ওয়ার' হিসেবে চিহ্নিত করা হয়। কোনো একটি প্রক্রিয়া ভাষার সংকলকটি বা 'কমপাইলারটি' এক ধরনের সিসটেম সফ্‌ট্‌ওয়ার। এই সংকলক প্রোগ্রামটি থাকায় ওই প্রক্রিয়া ভাষায়

লেখা প্রোগ্রাম কমপিউটার বুঝতে পারে। এখানে বিভিন্ন ধরনের সিসটেম সফ্‌ট্‌ওয়ার সম্বন্ধে আলোচনা করা হবে।

প্রতীকী ভাষান্তরক (Assembler) এবং সংকলক (Copmpiler) প্রোগ্রামগুলিকে সিসটেম সফ্‌ট্‌ওয়ার বলা হয়ে থাকে। এর কারণ এই প্রেগ্রামগুলি থাকায় কমপিউটারকে সহজভাবে ব্যবহার করা যায়। অর্থাৎ ওই সব ভাষায় প্রোগ্রাম লেখা সহজ হওয়াতে, অনেকের পক্ষেই প্রোগ্রাম লেখা সম্ভব। ফলে কমপিউটারের ব্যবহার বেড়ে যায়। এখন সফ্‌ট্‌ওয়ারের কাজই হচ্ছে কমপিউটারের ব্যবহার সহজ করে দেওয়া—যাতে এর ব্যবহার বেড়ে চলে। সেইজন্যে প্রতীকী ভাষান্তরক এবং সংকলক প্রোগ্রামগুলিকে সিসটেম সফ্‌ট্‌ওয়ার বলা হয়।

যে কোনো ধরনের ভাষান্তরক (সংকলক অথবা প্রতীকী ভাষান্তরক) ওই ভাষায় লেখা প্রোগ্রামকে যন্ত্রের ভাষায় রূপ দেয়। কিন্তু প্রোগ্রামটিকে যন্ত্রের ভাষায় রূপান্তরিত করার পরেই কমপিউটার সরাসরি প্রোগ্রামটির নির্দেশগুলি অনুসারে কাজ শুরু করতে পারে না। ভাষান্তরক যখন কোনো একটি প্রোগ্রামকে যন্ত্রের ভাষায় রূপ দেয় তখন কমপিউটার কোন স্মৃতিকোষ থেকে ওই যন্ত্রের ভাষায় নির্দেশগুলি রাখতে আরম্ভ করবে তা নির্দ্ধারণ করা যায় না। এর প্রধান কারণ অনেক সময়েই কতগুলি ছোট ছোট প্রোগ্রাম নিয়ে একটি বড় প্রোগ্রাম তৈরি হয়। ওই সব ছোট ছোট প্রোগ্রামগুলি আলাদা আলাদা ভাবে যন্ত্রের ভাষায় পরিবর্তন করা যায়। এখন আলাদা ভাবে যন্ত্রের ভাষায় রূপান্তর হওয়াতে আগের প্রোগ্রামটির নির্দেশগুলি রাখতে স্মৃতির কোন কোষ পর্যন্ত ব্যবহৃত হয়েছে সেই তথ্য মনে রেখে ভাষান্তরকটির পক্ষে এর পরের প্রোগ্রামের নির্দেশগুলি পরের কোন কোষ থেকে রাখবে তা স্থির করা সম্ভব নয়। ভাষান্তরক সব সময়েই প্রত্যেকটি ছোট প্রোগ্রামকেই রূপান্তর করার সময় একটি নির্দিষ্ট স্মৃতিকোষ ধরে নিয়ে সেই অনুসারে প্রোগ্রামটির নির্দেশগুলি রেখে যায়। এরপর একটি সিসটেম সফ্‌ট্‌-ওয়ার ওই সব ছোট ছোট যন্ত্রের ভাষায় রূপান্তরিত প্রোগ্রামগুলি সঠিকভাবে জোড়া দিয়ে স্মৃতিতে কোন কোন কোষে থাকবে তা ঠিক ক'রে স্মৃতিতে সব কটি প্রোগ্রামের নির্দেশগুলিই নিয়ে আসার ব্যবস্থা করে। ওই সিসটেম সফ্‌ট্‌ওয়ার-টিকে 'লিনকার' (Linker) এবং 'লোডার' (Loader) বলে। অর্থাৎ ওই সফ্‌ট্‌ওয়ারটির কাজ, আগের প্রোগ্রামটির নির্দেশ যে সব স্মৃতিকোষে আছে পরের প্রোগ্রামের নির্দেশ যাতে সেই একই স্মৃতিকোষে না রাখা হয়, তা ঠিকভাবে লক্ষ্য করা। কিন্তু পরের নির্দেশগুলি এমন ভাবে রাখতে হবে যাতে আগের প্রোগ্রামের নির্দেশ করার পরই পরের নির্দেশ অনুসারে কাজ করতে কোনোরকম ভুল না হয়।

কমপিউটারে কোনো সমস্যা সমাধানের জন্যে নানা ধরনের নির্দেশ থাকে, যেমন, দুটি সংখ্যার যোগ বা বিয়োগ, অনুপ্রবেশ অংশের সাহায্যে কোনো তথ্য কমপিউটারের স্মৃতিতে আনা বা স্মৃতিকোষের সঞ্চিত তথ্য নির্গম অংশের সাহায্যে লোকচক্ষুর অবগত করা। এইসব ভিন্ন ভিন্ন ধরনের কাজের জন্য কমপিউটারের এক একরকম সময় লাগে। সাধারণত যে সব কাজ করতে অনুপ্রবেশ বা নির্গম অংশের সাহায্য দরকার, সে সব কাজে অনেক বেশি সময়ের দরকার কারণ অনুপ্রবেশ এবং নির্গম অংশ মেকানিকাল যন্ত্রাংশ যুক্ত এবং সেইজন্য এতে বেশি সময় লাগে। কিন্তু সিপিউতে ইলেক-ট্রনিক্স যন্ত্রাংশ থাকায় এর সাহায্যে কোনো কাজ করতে তুলনায় কম সময় দরকার। একটি তথ্য অনুপ্রবেশ অংশের সাহায্যে স্মৃতিতে নিয়ে আসতে যে সময়ের প্রয়োজন সে সময়ে কমপিউটারের সিপিউ চুপচাপ বসে থাকলে কয়েক হাজার যোগ-বিয়োগ করার সময় নষ্ট হয়। কোনো তথ্য স্মৃতিতে আনার বা স্মৃতি থেকে নিয়ে যাওয়ার জন্য যখন অনুপ্রবেশ এবং নির্গম অংশের সাহায্য নেওয়া হয় তখন যাতে সিপিউ চুপচাপ বসে না থাকে তার জন্য ষাটের দশকে একটি নুতন হার্ডওয়ারের আবিষ্কার হয়। এর নাম 'আই/ও চ্যানেল' (I/O Channel) বা 'আই/ও প্রোসেসর'। যখন অনুপ্রবেশ বা নির্গম অংশের প্রয়োজন, আই/ও প্রোসেসার' কে নির্দেশ পাঠিয়ে দেওয়া হয় কাজটি করার জন্য। 'আই/ও প্রোসেসার' যতক্ষণ কাজটি করতে থাকবে ততক্ষণ সিপিউ অন্য প্রোগ্রামের অনেক কাজ করে ফেলতে পারে, কারণ আই/ও প্রোসেসারের কাজটি করার জন্য সিপিউ-এর দরকার নেই। এর ফলে সিপিউকে আর চুপচাপ বসে থাকতে হয় না। সেইজন্যে সময় বেঁচে যায়। কিন্তু 'আই/ও প্রোসেসার'-এর কাজটি সম্পন্ন হলেই সিপিউ-ওই প্রোগ্রামটির নির্দেশগুলি ধরে আবার কাজ আরম্ভ করবে। ওই সুবিধেটুকু যে সিসটেম সফ্ট্ওয়ারের সাহায্যে করা হয়ে থাকে তার নাম 'আই ও সি এস' (IOCS–Input Output Control System)। 'আই/ও চ্যানেল' একটি হার্ডওয়ার, কিন্তু 'আই ও সি এস' একটি সফ্ট্ওয়ার। হার্ডওয়ারটি না থাকলে সিপিউকে সময় নষ্ট করতে হত। আবার হার্ডডয়ারটি আছে, কিন্তু 'আই ও সি এস' প্রোগ্রামটির অভাবেও সিপিউ-এর সময়ের অপব্যবহার হবে।

একটি কমপিউটারে নানা ভাষায় লেখা প্রোগ্রাম চলা সম্ভব, যেমন সেই কমপিউটারের যন্ত্রের ভাষা, প্রতীকী ভাষা, আবার নানা ধরনের প্রক্রিয়া ভাষা–ফরট্রান, কোবল, বেসিক ইত্যাদি হতে পারে। যখন কমপিউটারে কোনো একটি প্রতীকী ভাষায় বা প্রক্রিয়া ভাষায় লেখা প্রোগ্রাম চালানো হয় তখন প্রোগ্রামটি কোন ভাষায় লেখা

নির্দেশগুলির আগে কয়েকটি নির্দেশ কোনো একটি বিশেষ ভাষায় দেওয়া হয়। ওই ভাষা আবার এক এক ধরনের কমপিউটারে এক একরকম। এই ধরনের নির্দেশে বলা থাকে, এরপর যে প্রোগ্রামটি আছে তার নির্দেশগুলি কোন ভাষায় লেখা এবং সেই অনুসারে ওই ভাষার ভাষান্তরকটি বা সংকলকটি কমপিউটারের স্মৃতিতে আনা হয়। ওই নির্দেশ কটি বুঝে ঠিকমত কাজ করার জন্য একটি সিসটেম সফ্‌ট্‌ওয়ার আছে যেটি সাধারণত 'অপারেটিং সিসটেম' (Operating System) বলে চিহ্নিত। একে অনেক সময়েই 'সুপার-ভাইজার' (Supervisor), 'মনিটর' (Monitor) বা 'একসিকিউটিভ' (Executive) নামেও আখ্যা দেওয়া হয়। ওই অপারেটিং সিসটেমের একটি অংশ সব সময়েই কমপিউটারের 'রম' (ROM) স্মৃতিতে অবস্থান করে। বাকি অংশ ডিস্‌ক, টেপের মত স্মৃতি সহায়ক কোনো কিছুতে রাখা হয়। প্রয়োজন হলে কমপিউটারের স্মৃতিতে ওই বাকি অংশটি আনা হয়ে থাকে।

কোনো একটি প্রোগ্রাম কমপিউটারে দেওয়ার সঙ্গে সঙ্গে অপারেটিং সিসটেম ওই নির্দেশগুলির পরের প্রোগ্রামটি কোন ভাষাতে লেখা, তা ওই প্রোগ্রামটির আগের নির্দেশগুলি দেখে বুঝে নেয়। অপারেটিং সিসটেম তখন সেই ভাষার সংকলকটি স্মৃতিতে আনার ব্যবস্থা করে। এরপর সংকলকটি প্রোগ্রামের নির্দেশগুলি সঠিক কিনা দেখে নেয়। যদি তা সঠিক থাকে তবেই যন্ত্রের ভাষায় সেগুলি সে পরিবর্তন করবে। এখন যন্ত্রের ভাষায় নির্দেশগুলির রূপান্তর করার পরে যদি প্রোগ্রামটির সাহায্যে তখনই সমস্যাটি সমাধানের প্রয়োজন হয় তা হলে অপারেটিং সিসটেম আবার 'লিনকার' এবং 'লোডার'কে নির্দেশ পাঠায়। এরা তখন যন্ত্রের ভাষায় লেখা প্রোগ্রমাটি কমপিউটারে স্মৃতিতে রাখার ব্যবস্থা করে এবং প্রোগ্রামটির নির্দেশগুলি অনুসারে কাজ শুরু করে দেয়। তবে কাজ চলার সময়ে যদি অনুপ্রবেশ বা নির্গম অংশের সাহায্যের দরকার হয় তবে আবার অপারেটিং সিসটেমের ডাক পড়ে। অপারেটিং সিসটেম তখন 'আই/ও প্রোসেসার' কে কাজটি করার নির্দেশ দেয়। সেই সঙ্গে সিপিউ-এর সাহায্যে অন্য কোনো প্রোগ্রামের নির্দেশগুলি আরম্ভ করার নির্দেশও পাঠায়। অপারেটিং সিসটেম একই সময়ে একের বেশি প্রোগ্রামের কাজ চলতে সাহায্য করে। যখন এক সঙ্গে একটির বেশি প্রোগ্রামের কাজ চলে তখন তাকে 'মালটিপ্রোগ্রামিং' (Multiprogramming) বলা হয়। এদিকে 'আই/ও প্রোসেসার'-এর কাজটি শেষ হয়ে গেলেই অপারেটিং সিসটেমকে সংকেত পাঠায়। সংকেত পাওয়ার পর অপারেটিং সিসটেম আবার সিপিউতে যে প্রোগ্রামটি চলছিল তাকে থামিয়ে দিয়ে আগের প্রোগ্রামটির যে পর্যন্ত কাজ হয়েছিল তারপর থেকে

করার জন্য সিপিউকে নির্দেশ পাঠায়। অপারেটিং সিসটেমের সব সময়ে হিসেব রাখা দরকার, কোন প্রোগ্রামের কাজ কতদূর পর্যন্ত হয়েছে এবং সেইভাবে সব দিকে নজর রেখে কাজগুলি এগিয়ে নিয়ে যেতে হয়। একটি কমপিউটারে যে সব অংশ আছে এবং এর সঙ্গে যে সব যন্ত্র লাগানো থাকে প্রোগ্রামগুলির প্রয়োজন অনুসারে সে সব যন্ত্রাংশ সুষ্ঠুভাবে বন্টন করা অপারেটিং সিসটেমের কাজ। সিপিউ-এর সময়ের যেন কোনো অপচয় না ঘটে, অপারেটিং সিসটেমকে তা বিশেষ ভাবে লক্ষ্য রাখতে হয়। অর্থাৎ কোনো প্রোগ্রামের যদি টেপ দরকার হয় এবং সেই সময়ে যদি অন্য কোনো প্রোগ্রাম সেই টেপ ড্রাইভ ব্যবহার না করে তবে প্রোগ্রামটির টেপ ব্যবহারের ব্যবস্থা করা উচিত। কিন্তু যদি সেই সময়ে অন্য কোনো প্রোগ্রাম টেপ ড্রাইভটি ব্যবহার করতে থাকে, তাহলে যতক্ষণ না আগের প্রোগ্রামের কাজ শেষ হয় ততক্ষণ এই প্রোগ্রামটিকে অপেক্ষা করতে হয়। অবশ্য সেই সংস্থায় আর কোনো টেপ ড্রাইভ থাকলে অন্য কথা। আবার যদি কোনো প্রোগ্রামের অনুপ্রবেশ বা নির্গম অংশের প্রয়োজন হয় তবে সেই প্রোগ্রামটিকে তা দেওয়ার ব্যবস্থা করা দরকার এবং ওই সময়ের মধ্যে অন্য কোনো প্রোগ্রামের কিছু সংখ্যক নির্দেশ সিপিউটিকে দিয়ে করিয়ে নেওয়া উচিত যাতে সিপিউ চুপচাপ বসে না থাকে। অপারেটিং সিসটেমকে সব সুষ্ঠুভাবে বন্টন করা ছাড়াও কোন প্রোগ্রামের কাজ কতটা হয়েছে এবং এরপর কোন অংশের সাহায্যে কি করতে হবে সেসবের হিসেব নিখুঁতভাবে রাখতে হয়।

ফ্লো-চার্ট
বা
প্রবাহ চিত্র

কমপিউটারের সাহায্যে কোনো সমস্যা সমাধানের জন্য কমপিউটারের বোধগম্য যে কোনো ভাষায় প্রথমে একটি প্রোগ্রাম লেখা হয়ে থাকে। এই প্রোগ্রাম কতগুলি নির্দেশের সমষ্টি। নির্দেশগুলি প্রথমে স্মৃতিকোষে সঞ্চয় করা হয়। তা ছাড়া যে সব তথ্যের (data) সাহায্যে নির্দেশগুলি পালন করা হয় সেগুলিও স্মৃতিকোষে রাখার ব্যবস্থা থাকে। দৃষ্টান্ত হিসেবে যোগের কথা ধরা যাক। যোগের বেলায় নির্দেশ পালন করার জন্য স্বাভাবিকভাবেই দুটি সংখ্যার প্রয়োজন হবে। আমরা জানি, যে কোনো সমস্যা সমাধানের ক্ষেত্রে কমপিউটার নিজে থেকে চিন্তা-ভাবনা করে কোনো কাজ করতে পারে না। সেইজন্যই এই নির্দেশগুলি যথাযথ হওয়া দরকার। না হলে সমস্যাটির সঠিক সমাধান পাওয়া সম্ভব নয়। যথাযথ নির্দেশের অর্থ, এগুলি এমনভাবে সাজাতে হবে যাতে কোন নির্দেশের পর কোন নির্দেশটি পালন করতে হবে কমপিউটার যেন তা সরাসরি বুঝতে পারে। সঠিকভাবে পরপর সাজানো এই নির্দেশগুলির সাহায্যে যদি কোনো সমস্যার সমাধান পাওয়া যায় তবে সেই নির্দেশগুলিকে কমপিউটারের ভাষায় 'অ্যালগরিদম্' (algorithm) বলা হয়। এই অ্যালগরিদম্ কমপিউটারের তিন ধরনের ভাষার (যন্ত্রের ভাষা, ওই যন্ত্রের প্রতীকী ভাষা বা অ্যাসেম্বলী ভাষা, কোনো একটি প্রক্রিয়া ভাষা বা উচ্চ-পর্যায়ের ভাষা) যে কোনো এক ধরনের ভাষাতে লেখা যেতে পারে। কিন্তু এই তিন ধরনের ভাষার কোনো একটি ভাষায় লেখার আগে এই অ্যালগরিদম্‌টি ঠিকমত কাজ করবে কিনা সাধারণত তা একটি 'ফ্লো-চার্ট' (flow chart) বা প্রবাহ চিত্রের সাহায্যে দেখে নেওয়া

হয়। এই ফ্লো-চার্টে একটি অ্যালগরিদম্‌কে একটি চিত্রের সাহায্যে বোঝানো হয়ে থাকে। এই চিত্রে এক এক ধরনের নির্দেশের জন্য এক এক রকমের নকশার সাহায্য নেওয়া হয়। গৃহ-নির্মাণের পূর্বে তার যেমন একটা নকশা অপরিহার্য, যাতে বাড়িতে কোথায় কোন ধরনের ঘর থাকবে তা যেমন বোঝা যায় সহজে, তেমনি একটি অ্যালগরিদম্‌কেও কমপিউটারের বোধ্য ভাষায় না লিখে প্রথমে বাড়ির নকশার মত এটির একটি প্রবাহ চিত্র তৈরি করা হয়। কমপিউটার অবশ্য প্রবাহ চিত্র বুঝতে পারে না, কিন্তু এর সুবিধে, কোন নির্দেশের পর কোন নির্দেশটি একে পালন করতে হবে তার একটি পরিষ্কার ছবি এতে ফুটে ওঠে। ফলে সমস্যাটি সমাধানের জন্য নির্দেশগুলি সঠিকভাবে সাজানো আছে কিনা, তা এ থেকে বুঝে নেওয়া যায়। যদি নির্দেশগুলি সঠিকভাবে সাজানো থাকে তাহলে এর থেকে সরাসরি কোনো একটি উচ্চ পর্যায়ের ভাষায় অ্যালগরিদম্‌টি লিখে কমপিউটারে চালিয়ে সমস্যাটির সমাধান করা সম্ভব।

কোনো একটি সমস্যা সমাধানের প্রবাহ চিত্র থাকলে নতুন কারও পক্ষে সেটি তাড়াতাড়ি সহজে বুঝে নেওয়া সম্ভব এবং প্রবাহ চিত্রটি একটি প্রামাণ্য চিত্র হিসেবে রেখে দেওয়া চলে। প্রবাহ চিত্র সাধারণত দুরকমের, সিসটেম ফ্লো-চার্ট এবং প্রোগ্রাম ফ্লো-চার্ট।

একটি সংস্থার সমস্ত কাজের প্রবাহ সম্বন্ধে ধারণা করার জন্য সিসটেম ফ্লো-চার্টের ব্যবহার হয়। অর্থাৎ এই ধরনের প্রবাহ চিত্রের সাহায্যে একটি সংস্থায় কোন কাজের পর কোন কাজটি করা প্রয়োজন তার একটি পরিষ্কার ছবি ফুটিয়ে তোলা সম্ভব। এই চিত্রে যে সমস্ত কাজের কথা বলা থাকে সে সবই যে কমপিউটারেই করতে হবে, তা নাও হতে পারে।

দ্বিতীয় ধরনের পবাহ চিত্র ব্যবহার করা হয় কখন ? কমপি-উটারের সাহায্যে কোনো কাজ সমাধান করার সময়ে কোন নির্দেশের পর কোন নির্দেশটি পালন করতে হবে তা বোঝানোর জন্য দ্বিতীয় ধরনের প্রবাহ চিত্রের ব্যবহার। অর্থাৎ সিসটেম ফ্লো-চার্টে কেবলমাত্র বলা থাকছে কমপিউটারের সাহায্যে এই কাজটি করতে হবে আর প্রোগ্রাম ফ্লো-চার্টে সেই কাজটি সমাধানের জন্য যা যা করা প্রয়োজন সে সবেরই নির্দেশ থাকবে।

দুধরনের প্রবাহ চিত্রেই ভিন্ন ভিন্ন ধরনের কাজের জন্য বিভিন্ন রকমের নকশা ব্যবহার করা হয়। তবে এখানে কেবলমাত্র 'প্রোগ্রাম ফ্লো-চার্ট' সম্বন্ধেই আলোচনা করা হবে এবং 'প্রোগ্রাম ফ্লো-চার্ট'-এর বদলে সংক্ষেপে ফ্লো-চার্ট বা প্রবাহ চিত্র হিসেবেই তাকে উল্লেখ করা হবে।

এবারে প্রবাহ চিত্রে ভিন্ন ভিন্ন ধরনের নির্দেশের জন্য যেসব নকশার ব্যবহার করা হয় সে বিষয়ে আলোচনা করবো।

1. তীর চিহ্ন (→)— এই চিহ্নের সাহায্যে এর পরে কোন নির্দেশটি অনুসরণ করা হবে তা বোঝানো হয়।
2. উপবৃত্তাকার ক্ষেত্র (⬭)— প্রবাহ চিত্র শুরু এবং শেষ বোঝাতে এই চিহ্নটি ব্যবহার করা হয়।
3. সামান্তরিক ক্ষেত্র (▱)— অনুপ্রবেশ অংশের সাহায্যে কোনো তথ্য স্মৃতিতে নিয়ে আসতে হলে অথবা স্মৃতিকোষে রাখা কোনো তথ্য নির্গম অংশের মাধ্যমে ছাপালে এই ধরনের নকশা ব্যবহার করা হয়। এই নকশাটিতে কেবলমাত্র একটিই নির্গমন পথ থাকে।
4. আয়তাকার ক্ষেত্র (▭)— যোগ, বিয়োগ, গুণ, ভাগ দ্বারা নির্ণীত ফলটি একটি স্মৃতিকোষে রাখার জন্য এই নকশাটি ব্যবহার করা হয়। এই ক্ষেত্র থেকেও কেবলমাত্র একটি নির্গমন পথ বেরোনো সম্ভব।
5. হীরকাকৃতি ক্ষেত্র (◇)— সিদ্ধান্ত নেওয়ার নির্দেশের জন্য এই ধরনের নকশার, ব্যবহার করা হয়। এর একাধিক নির্গমন পথ থাকে।
6. বৃত্তাকার ক্ষেত্র (○)— প্রবাহ চিত্রের একটি অংশের সঙ্গে অপর একটি অংশের যোগাযোগ করার জন্য এই ধরনের নকশার ব্যবহার করা হয়ে থাকে। অনেক সময়েই একটি অ্যালগরিদমের পুরো ছবিটি একটি পাতায় ধরানো সম্ভব না হতেও পারে। সেক্ষেত্রে পরের পাতার সঙ্গে যোগাযোগ করার জন্য এই নকশাটির ব্যবহার করা হয়।

এবারে কয়েকটি সমস্যা প্রবাহ চিত্রের সাহায্যে সমাধান করে দেখানো হবে।

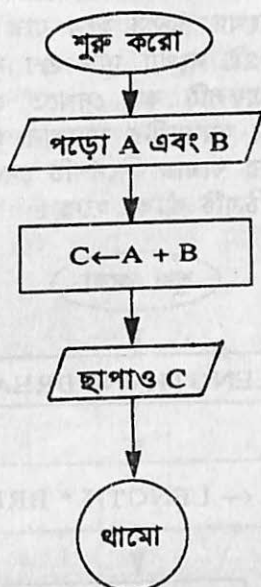
উদাহরণ 1.

দুটি সংখ্যার যোগফল বের করতে হবে।

সমাধান ঃ

দুটি সংখ্যাকে কমপিউটারের সাহায্যে যোগ করতে হলে প্রথমে সংখ্যা দুটি অনুপ্রবেশ অংশের সাহায্যে স্মৃতিকোষে সঞ্চয় করতে হবে। কারণ কমপিউটারের সঙ্গে বাইরের যোগাযোগ অনুপ্রবেশ এবং নির্গম অংশের সাহায্যে করা হয়ে থাকে। কিন্তু সংখ্যা দুটি অনুপ্রবেশ অংশে যোগ করা যায় না আবার সরাসরি সিপিউতেও নিয়ে যাওয়াও সম্ভব নয়। সেইজন্য দুটি সংখ্যাকে প্রথমে স্মৃতিকোষে সঞ্চয় করতে হয়। এরপর সংখ্যা দুটি যোগ করে যোগফলটি অপর একটি স্মৃতিকোষে রাখা হয়। তবে যোগফলটি দেখার দরকার হলে তা নির্গম অংশের সাহায্য নিয়ে দেখা চলে।

এজন্যেও নির্দেশের প্রয়োজন এবং সবশেষে থামার জন্যও নির্দেশ দিতে হবে। যদি থামার নির্দেশ না থাকে তাহলে স্মৃতিকোষে যা থাকবে সেই অনুসারে সে কাজ করবে।

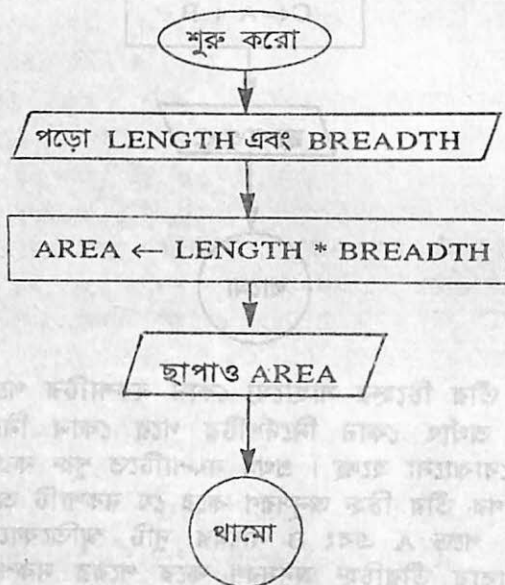
এবারে প্রবাহ চিত্রের সাহায্যে অ্যালগরিদমটিকে দেখানো হচ্ছে।

এখানে তীর চিহ্নের সাহায্যে কোন নকশাটির পরে কোনটিতে যেতে হবে অর্থাৎ কোন নির্দেশটির পরে কোন নির্দেশটি পালন করবে তা বোঝানো হচ্ছে। প্রথম নকশাটিতে শুরু করার কথা বলা আছে। এরপর তীর চিহ্ন অনুসরণ করে যে নকশাটি আসছে সেখানে দুটি সংখ্যা পড়ে A এবং B নামের দুটি স্মৃতিকোষে তা রেখে দিচ্ছে। আবার তীরচিহ্ন অনুসরণ করে পরের নকশাটিতে সংখ্যা দুটি যোগ করে যোগফলটি C নামের স্মৃতিকোষে রেখে দিতে বলা হচ্ছে। এই নকশাটিতে ব্যবহৃত তীর চিহ্নের অর্থ, C নামের স্মৃতিকোষে আগে যদি কোনো সংখ্যা থাকে তবে তার বদলে এই নতুন যোগফলটি রাখা হবে। এরপর তীরচিহ্ন অনুসরণ করে C-এর সংখ্যাটি প্রিন্টারের সাহায্যে ছাপানো হবে এবং সবশেষের নকশাটির নির্দেশ অনুসারে থামবে অর্থাৎ সমস্যাটির সমাধানের জন্য এখন আর কিছুই করার নেই।

উদাহরণ 2.

নিম্নের সূত্র অনুসারে একটি আয়তাকার ক্ষেত্রের ক্ষেত্রফল বের করতে হবে।

[ক্ষেত্রফল = দৈর্ঘ্য × প্রস্থ]

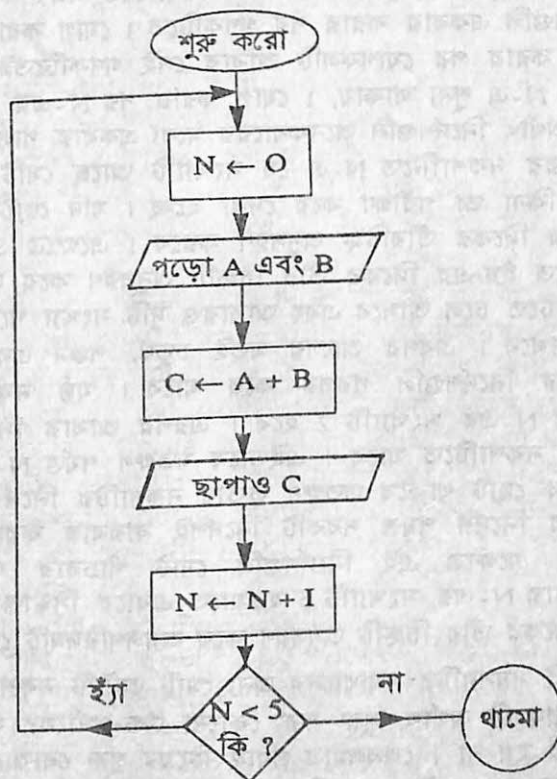
সমাধান ঃ

একটি আয়তাকার ক্ষেত্রের ক্ষেত্রফল বের করার প্রয়োজন হলে প্রথমে ওই ক্ষেত্রের দৈর্ঘ্য এবং প্রস্থ জানতে হবে। অর্থাৎ অনুপ্রবেশ অংশের মাধ্যমে দুটি সংখ্যা পড়তে হবে যার একটি হবে দৈর্ঘ্য এবং অন্যটি প্রস্থ। এরপর ওই সংখ্যা দুটি গুণ করে ক্ষেত্রটির ক্ষেত্রফল বের করা হবে। ক্ষেত্রফলটি কত দেখতে হলে এবারে ছাপানোর নির্দেশ দিতে হবে এবং সমস্যাটির সমাধান হয়ে গেছে বলে আগের উদাহরণের মতই এরপর থামার নির্দেশটি দেওয়া প্রয়োজন। এবারে এই সমস্যাটির প্রবাহ চিত্রটি আঁকা হচ্ছে।

উদাহরণ 1-এর মত এখানেও দুটি সংখ্যা পড়া হচ্ছে। এই সংখ্যা দুটি LENGTH এবং BREADTH নামের স্মৃতিকোষে রাখা হল। এই স্মৃতিকোষের নামকরণ এক একটি উচ্চ পর্যায়ের ভাষায় এক একরকম ভাবে করা হয়। তবে বেশির ভাগ ক্ষেত্রেই এমনভাবে নামকরণ করা হয় যার ফলে ওই স্মৃতিকোষে কি রাখা আছে তা বোঝা যায়। এখানেও LENGTH এবং BREADTH-এর সহায্যে বোঝানো হচ্ছে যে, প্রথম সংখ্যাটি দৈর্ঘ্য এবং অপরটি প্রস্থ। এরপরের নকশাটিতে ওই সংখ্যা দুটি গুণ করে ক্ষেত্রফল পাওয়া যাবে এবং তা AREA নামের স্মৃতিকোষে রাখা হবে। এরপর আগের উদাহরণের মতই ক্ষেত্রফল ছাপিয়ে থামতে বলা হচ্ছে।

উদাহরণ 3.

5 টি ভিন্ন জোড়া সংখ্যার যোগফলগুলি বের করতে হবে।

সমাধান ঃ

প্রশ্নটির সমাধান নানাভাবে করা যেতে পারে। একটি উপায়ে উদাহরণ 1-এর প্রবাহ চিত্র অনুসরণ করেও সমাধান করা সম্ভব। এতে কমপিউটারের কোনো একটি উচ্চ পর্যায়ের ভাষায় যে প্রোগ্রাম পাওয়া যাবে, সেটি পাঁচবার চালাতে হবে। কিন্তু এতে কমপিউটারের অনেকটা সময় লেগে যাবে। দ্বিতীয় উপায়, ওই প্রবাহ চিত্রটিকে এমনভাবে করা দরকার যাতে এর থেকে যে প্রোগ্রাম করা হবে কমপিউটারে তা একবার চালিয়েই সমস্যাটির সমাধান পাওয়া যাবে। এই নতুন প্রবাহ চিত্রটি এবারে দেখানো হচ্ছে।

এখানে উদাহরণ 1-এর প্রবাহ চিত্রের নির্দেশগুলি ছাড়া আরও তিনটি বেশি নির্দেশ লক্ষ্য করা যাচ্ছে। একই ধরনের কাজ বারবার করার জন্য এদের প্রয়োজন। বারবার বললে অবশ্য সঠিকভাবে

বোঝা যায় না ঠিক কতবার করতে হবে। কমপিউটারের সাহায্য নিতে গেলে ঠিক কতবার করা দরকার পরিষ্কার করে বলা প্রয়োজন। এছাড়া প্রত্যেকবার নির্দেশগুলি পালন করার পর যতবার করার কথা তা করা হয়েছে কিনা তা পরীক্ষা করেও দেখতে হবে। এরজন্য প্রথমে একটি স্মৃতিকোষকে গণক হিসেবে দেখানো হয়ে থাকে। ওই গণকটিতে প্রথমে একটি সংখ্যা রাখা হয়। সাধারণত এই সংখ্যাটি শূন্য। উপরের প্রবাহ চিত্রে এই শূন্য রাখার কাজ দ্বিতীয় নকশাটিতে করা হয়েছে। এখানে N গণক হিসেবে উপস্থাপিত। প্রথমে শূন্য রাখার কারণ আছে। এ থেকে বোঝা যাবে, যে নির্দেশগুলি বারবার করতে হবে এখনও পর্যন্ত তা একবারও করা হয় নি। এরপর প্রথম প্রবাহ চিত্রের মতই সংখ্যা দুটি পড়ে স্মৃতিকোষে রেখে, যোগ করে, যোগফলটি ছাপানো হল। এই কাজগুলি তৃতীয়, চতুর্থ এবং পঞ্চম নকশাটিতে লক্ষ্য করা যায়। এই নির্দেশগুলি একবার করার পর গণকটিতে 1 যোগ করা হল এবং তা যোগ করার পর যোগফলটি আবার সেই গণকটিতেই রাখা হল। প্রথমে N-এ শূন্য থাকায়, 1 যোগ করার পর N-এর সংখ্যাটি হবে 1। অর্থাৎ নির্দেশগুলি অনেকবারের মধ্যে একবার পালন করা হল। এরপরের নকশাটিতে N-এ যে সংখ্যাটি আছে সেটি 5-এর থেকে ছোট কিনা তা পরীক্ষা করে দেখা হচ্ছে। যদি ছোট হয় তা হলে হ্যাঁ-এর দিকের তীরচিহ্ন অনুসরণ করবে। এক্ষেত্রে প্রথমবার ছোট হওয়াতে হ্যাঁ-এর দিকের তীর চিহ্নটি অনুসরণ করে আবার তৃতীয় নকশাটিতে চলে আসবে এবং আবারও দুটি সংখ্যা পড়ে A এবং B-তে রাখবে। এরপর আগের মতই চতুর্থ, পঞ্চম এবং ষষ্ঠ নকশা তিনটির নির্দেশগুলি পরপর করে যাবে। ষষ্ঠ নকশাটিতে এসে এবারে N-এর সংখ্যাটি 2 হবে। এরপর আবার সিদ্ধান্ত অনুযায়ী তৃতীয় নকশাটিতে যাবে। এইভাবে যতক্ষণ পর্যন্ত N-এর সংখ্যাটি 5 থেকে ছোট থাকবে ততক্ষণ তৃতীয় নকশাটির নির্দেশ থেকে সপ্তম নকশার নির্দেশ পর্যন্ত সবকটি নির্দেশই বারবার করবে। বারবার অর্থ্যাৎ এক্ষেত্রে এই নির্দেশগুলি মোট পাঁচবার করতে হবে। পঞ্চমবার N-এর সংখ্যাটি 5 হওয়াতে এবারে সিদ্ধান্ত অনুযায়ী না-এর দিকের তীর চিহ্নটি অনুসরণ করে অ্যালগরিদমটি শেষ হবে।

এই সমস্যাটির সমাধানের জন্য মোট আটটি নকশার প্রয়োজন। তবে প্রথমটি অর্থাৎ 'শুরু কর' কোনো উচ্চ-পর্যায়ের ভাষায় লেখার দরকার হয় না। কেবলমাত্র প্রবাহ চিত্রের শুরু বোঝানোর জন্য এর ব্যবহার। কাজেই সমস্যাটির সমাধানের জন্য মোট সাতটি নকশা অর্থাৎ সাতটি নির্দেশের প্রয়োজন। এই সাতটি নির্দেশ প্রথমে সাতটি স্মৃতিকোষে (যদি এক একটি নির্দেশের জন্য একটি কোষ ধরা হয়) রাখা হবে। এছাড়া আরও চারটি কোষের প্রয়োজন—দুটির

প্রয়োজন সংখ্যা দুটির জন্য, একটি যোগফল এবং অপর একটি গণকের জন্য। অর্থাৎ মোট এগারোটি স্মৃতিকোষের দরকার হবে। এখানে বলে রাখা প্রয়োজন যে, দ্বিতীয়বার আবার দুটি সংখ্যা পড়ার সময়ে সেই সংখ্যা দুটির জন্য নতুন কোষের আর প্রয়োজন হবে না। প্রথমবার যে জায়গাগুলিতে সংখ্যা দুটি রাখা হয়েছিল সেই কোষ দুটিতেই এবারও রাখা হবে। এরপরের দুটি সংখ্যাকেও সেই আগের স্মৃতিকোষ দুটিতেই রাখা সম্ভব। কাজেই যত সংখ্যক দুটি সংখ্যাই যোগ করা হোক না কেন সেই এগারোটি কোষেরই প্রয়োজন। যদি একশত জোড়া সংখ্যা পড়ে, যোগ করে, যোগফলটি ছাপাতে হয় তাহলেও সাতটি নির্দেশ এবং মোট এগারোটি কোষেই কাজ চলবে। একশোটির ক্ষেত্রে আর সব নির্দেশই আগের মত থাকবে, কেবলমাত্র সপ্তম নকশাটির নির্দেশটিতে N<5-এর স্থানে N<100 লিখলেই ঠিক একশো বার দুটি করে সংখ্যা পড়ে যোগ করে যোগফলটি ছাপাবে।

এখানে একটি কথা মনে রাখা দরকার। এক ধরনের কাজ বারবার করতে হলে কমপিউটারের সাহায্যে তা করাই বেশি সুবিধেজনক। বারবার করার জন্য অতিরিক্ত খুব বেশি সংখ্যক নির্দেশের প্রয়োজন হয় না। কেবলমাত্র দুটি সংখ্যা যোগ করার জন্য কমপিউটার ব্যবহার করা উচিত হবে না, কিন্তু পাঁচশত দুটি সংখ্যা যোগ করার দরকার হলে কমপিউটার ব্যবহার করা যেতে পারে এবং এক্ষেত্রে কেবলমাত্র তিনটি বেশি নির্দেশ দিয়েই তা করা সম্ভব। এখানে একটা ব্যাপারে নজর রাখা প্রয়োজন। সিদ্ধান্তের পরে যে নকশাটিতে ফিরে যেতে বলা হচ্ছে তা সঠিক নকশা কিনা দেখা দরকার। যদি তা না হয় তাহলে নানা রকমের ভুল হওয়ার আশঙ্কা। দৃষ্টান্তস্বরূপ, উপরের প্রবাহ চিত্রে যদি 'হ্যাঁ'-এর দিকের তীর চিহ্নটি তৃতীয় নকশাটির পরিবর্তে দ্বিতীয় নকশাতে গিয়ে শেষ হয় তবে আর কখনোই 'না'-এর দিকের তীর চিহ্নের দিকে যাবে না। অর্থ্যাৎ N-এর সংখ্যাটি কোনো অবস্থাতেই 5 না হওয়াতে অ্যালগরিদম্‌টি আর শেষ হবে না এবং কমপিউটার না থেমে চলতেই থাকবে। এর কারণ কি ? প্রথমবার N-এ 1 থাকবে। পরের নির্দেশটি অনুসারে N<5 হওয়ার জন্যে 'হ্যাঁ'-এর দিকের তীর চিহ্ন অনুসরণ করে তা দ্বিতীয় নকশাটিতে আসবে। এখানে আসার পর N স্মৃতিকোষটিতে শূন্য রাখার জন্য এই নির্দেশটিতে বলা হচ্ছে। কাজেই N-এর 1 মুছে গিয়ে আবার শূন্য হবে। এরপর আবার দুটি সংখ্যা পড়ে, যোগ করে যোগফলটি ছাপানোর পর N-এ আবার 1 থাকবে। কাজেই সিদ্ধান্ত অনুযায়ী আবার দ্বিতীয় নকশাটিতে আসবে এবং N-এ শূন্য হবে। সুতরাং বারবার N-এ শূন্য এবং 1 হতে থাকবে। সেইজন্য কখনই N-এর সংখ্যাটি 5-এর সমান হবে

না এবং 'না'-এর দিকের তীর চিহ্নের দিকেও আর যাওয়া যাবে না। ফলে সামান্য একটু ভুলের জন্য প্রবাহ চিত্রটি ঠিকমত কাজ করবে না। সুতরাং প্রবাহ চিত্র করার সময়ে খুব ভালভাবে চিন্তা করে নজর দিয়ে করতে হবে। সেইসঙ্গে প্রবাহ চিত্র তৈরি হয়ে যাওয়ার পরে সমস্যার সমাধান ঠিকমত হচ্ছে কিনা কিছু নমুনার সাহায্যে তা দেখে নিতে হবে।

বারবার কিছু সংখ্যক নির্দেশ করার জন্য কয়েকটি অত্যাবশ্যক কথা মনে রাখা দরকার।

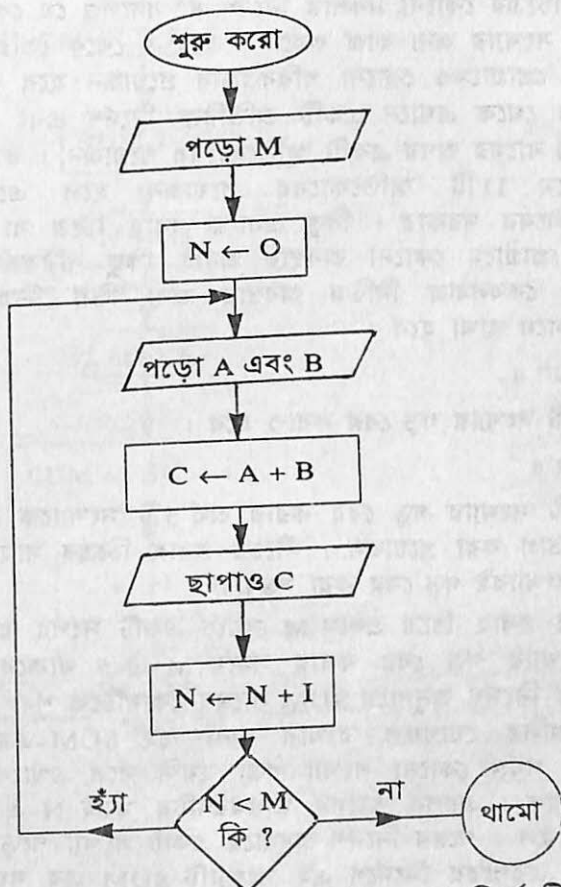
1. এক্ষেত্রে সাধারণত একটি গণকের প্রয়োজন। গণকটিতে প্রথমে একটি সংখ্যা রাখা হয় এবং সাধারণত তা শূন্য। বারবার যে সব নকশা করা হবে গণকটিতে শূন্য রাখার নির্দেশের নকশাটি সব সময়ই তার আওতার বাইরে থাকে। উপরের উদাহরণটিতে দ্বিতীয় নকশাটিকে বাইরে রাখা আছে এবং N-কে এখানে গণক হিসেবে ব্যবহার করা হয়েছে।

2. এরপর নির্দিষ্ট নির্দেশগুলি একবার করার পর গণকটিতে 1 যোগ করে যোগফলটি আবার সেই গণকটিতেই রাখতে হবে। উপরের উদাহরণে দুটি সংখ্যা একবার পড়ে, যোগ করে যোগফলটি ছাপানের পর N গণকটির সঙ্গে 1 যোগ করে আবার N-এই রাখা হচ্ছে। সব সময়েই যে সব নকশা বারবার করা হবে এই নকশাটি তার একটি।

3. গণকটিতে যে নতুন সংখ্যাটি রাখা হল পরীক্ষা করে দেখতে হবে নির্দেশগুলি যতবার করার কথা তা করা হয়েছে কিনা, যদি তা না হয় তবে 'না'-এর চিহ্নের দিকে গিয়ে আবার নির্দেশগুলির কাজ করবে। কিন্তু 'হ্যাঁ' হলে তার আর প্রয়োজন নেই। যেসব নকশা বারবার করতে হয় এই নকশাটিও তার একটি।

উপরের নির্দেশ তিনটি মেনে চললে অসীমবার হওয়ার সম্ভাবনা থাকে না। তবে বারবার কতগুলি নির্দেশ করার এটাই একমাত্র উপায় নয়। অন্যভাবেও তা করা সম্ভব। অপর একটি উপায় এর পরের একটি প্রবাহ চিত্রে করে দেখানো হবে।

উদাহরণ 3-এ কত জোড়া সংখ্যা নেওয়া হবে তার উপরে নির্ভর করে সপ্তম নির্দেশটি দিতে হবে। যদি সংখ্যা হয় পাঁচ জোড়া তবে নির্দেশটি হবে 'N<5 কি ?'। আবার একশো জোড়া হলে ওই নির্দেশটি হবে 'N<100 কি ?'। কাজেই প্রবাহ চিত্রটিতে ওই সপ্তম নির্দেশটি প্রয়োজনে বদলাতে হবে। এটি খুব যুক্তিযুক্ত নয়। কোনো একটি অ্যালগরিদমের ফ্লো-চার্ট বা প্রবাহ চিত্র থেকে কমপিউটারের ভাষায় যখন একটি প্রোগ্রাম লেখা হয় তখন সেটি এমন হওয়ার দরকার যাতে কোনো অবস্থাতেই প্রোগ্রামে কোনো পরিবর্তনের

প্রয়োজন না হয়। এবারে উপরের প্রবাহ চিত্রে আর একটি নকশা জুড়ে দেখানো হচ্ছে যে কোনো অবস্থাতেই ওই প্রবাহ চিত্রের কোনো নির্দেশেরই পরিবর্তনের প্রয়োজন হবে না।

এবারের প্রবাহ চিত্রের দ্বিতীয় নকশাটিতে যে নির্দেশটি আছে তা একটি সংখ্যা পড়ে M নামের স্মৃতিকোষে রাখছে। এই সংখ্যাটি এক একবার এক একরকম হবে। যদি সংখ্যা হয় 5 জোড়া, তবে M-এ 5 থাকবে, আবার 100 জোড়া সংখ্যার বেলায়, M স্মৃতিকোষে 100 রাখতে হবে। এরপর অষ্টম নকশাটিতে (আগের প্রবাহ চিত্রের সপ্তম নকশা) N-এর সঙ্গে M-এ যে সংখ্যাটি রাখা হয়েছে তা তুলনা করে দেখা হচ্ছে। M-এ 5 থাকলে যতক্ষণ পর্যন্ত N-এর সংখ্যা 5-এর ছোট ততক্ষণ নির্দেশগুলি কাজ করতে থাকবে। আবার M-এ যদি 100 থাকে তবে N-এর সংখ্যা 100-এর থেকে ছোট থাকা পর্যন্ত নির্দেশগুলি কাজ করবে। এক্ষেত্রে অষ্টম নকশাটির

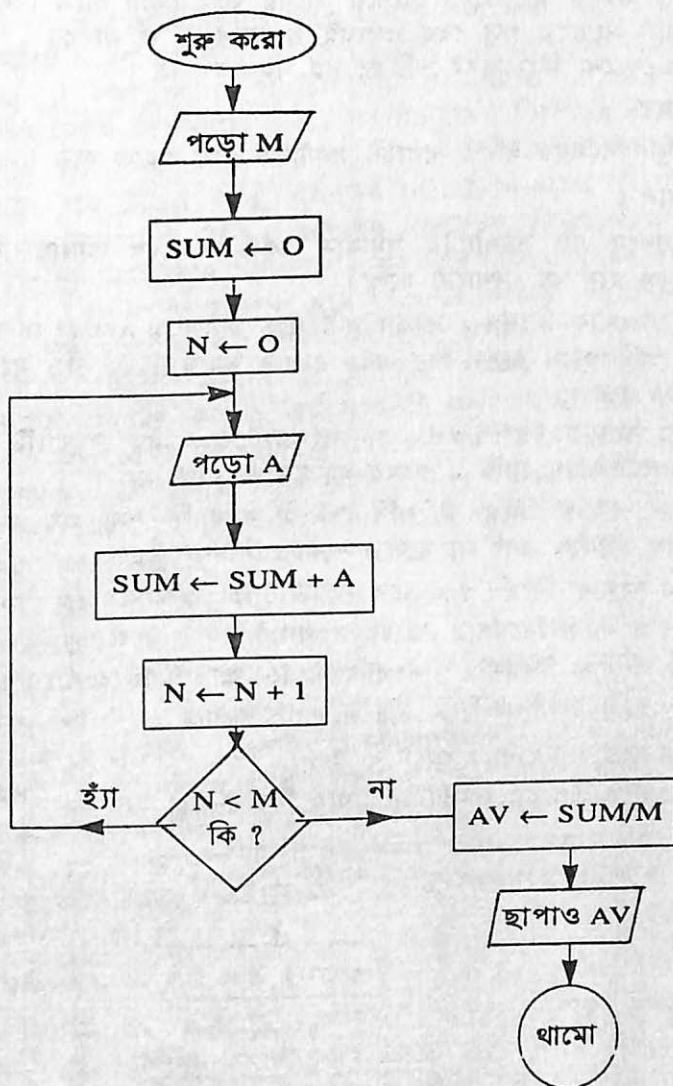
নির্দেশ 5-এর বেলাতে একরকম, আবার 100-এর বেলায় যে আর একরকম হবে, তা নয়। দুটি ক্ষেত্রেই একই নির্দেশ অর্থাৎ N, M-এর সংখ্যাটির থেকে ছোট কিনা তা দেখা হচ্ছে। কাজেই এই প্রবাহ চিত্রের কোনো নকশার নির্দেশ না বদলেও যে কোনো সংখ্যক জোড়া সংখ্যার জন্য কাজ করবে। আর এ থেকে তৈরি কমপিউটার ভাষার প্রোগ্রামেও কোনো পরিবর্তনের প্রয়োজন হবে না। আগের প্রোগ্রাম থেকে এখানে একটি অতিরিক্ত নির্দেশ এবং তথ্য রাখার জন্য M নামের অপর একটি স্মৃতিকোষের প্রয়োজন। কাজেই আগের প্রোগ্রামে 11টি স্মৃতিকোষের প্রয়োজন হলে এক্ষেত্রে 13টি স্মৃতিকোষের দরকার। কিন্তু এবারে প্রবাহ চিত্রে বা তার থেকে তৈরি প্রোগ্রামে কোনো অবস্থার জন্যই কিছু পরিবর্তনের দরকার নেই। কেবলমাত্র বিভিন্ন অবস্থার জন্য ভিন্ন ভিন্ন সংখ্যা M স্মৃতিকোষে রাখা হবে।

উদাহরণ 4.

5টি সংখ্যার গড় বের করতে হবে।

সমাধান ঃ

5টি সংখ্যার গড় বের করার অর্থ 5টি সংখ্যাকে যোগ করে 5 দিয়ে ভাগ করা প্রয়োজন। নীচের প্রবাহ চিত্রের সাহায্যে যতগুলি ইচ্ছা সংখ্যারই গড় বের করা সম্ভব।

এই প্রবাহ চিত্রে প্রথমে M কোষে একটি সংখ্যা রাখা হল। 5 টি সংখ্যার গড় বের করার সময়ে M-এ 5 থাকবে। এরপরের নকশার নির্দেশ অনুসারে SUM নামের কোষটিতে শূন্য রাখা হবে। সংখ্যাগুলির যোগফল রাখার জন্য এই SUM-এর প্রয়োজন। এখনও পর্যন্ত কোনো সংখ্যা পড়া হয়নি বলে এখানে প্রথমে শূন্য রাখা হল। এরপর আগের উদাহরণটির মতই N-এ শূন্য রাখার ব্যবস্থা হল। পরের নির্দেশ অনুসারে একটি সংখ্যা পড়ে A-তে রাখা হবে। এরপরের নির্দেশে এই সংখ্যাটি SUM-এর সংখ্যাটির সঙ্গে যোগ করার পর যোগফলটি SUM-এই রাখতে হবে। প্রথম বারে SUM-এ আগে শূন্য থাকায় এবারে SUM-এ প্রথম সংখ্যাটিই থাকবে। কিন্তু পরের বারে দ্বিতীয় সংখ্যাটি যোগ করার সময়ে SUM-এ প্রথম সংখ্যাটি থাকায় প্রথম দুটি সংখ্যার যোগফল SUM-এ থাকবে। এইভাবে এক এক করে সংখ্যাগুলি পড়ে আগের যোগফলের সঙ্গে নতুন আনা সংখ্যাটি যোগ করে যোগফলটি SUM-এ রাখা হচ্ছে। পঞ্চমবারে সংখ্যাটি পড়ে যখন 'N<M কি ?' এই নির্দেশে আসবে তখন SUM-এ পাঁচটি সংখ্যার যোগফল এবং N-এ 5 থাকবে। কাজেই সিদ্ধান্ত অনুসারে না-এর দিকের তীরচিহ্ন অনুসরণ করে SUM-এর যোগফলকে M-এর সংখ্যা দিয়ে ভাগ

করে AV-তে গড় রাখা হবে। এরপর এই গড় ছাপাবার পর থামার নির্দেশ অনুসারে থেমে যাবে। এখন 5 টি সংখ্যার বদলে যদি 20টি সংখ্যার গড় বের করতে হয় তবে M-এ 20 রাখা হবে এবং এরপর 20 টি সংখ্যা এক এক করে পড়ে SUM-এ যোগ করা হবে। সবকটি সংখ্যা পড়ে, যোগ করার পর যোগফলকে M-এর সংখ্যা অর্থাৎ এবারে 20 দিয়ে ভাগ করে 20টি সংখ্যার গড় বের করা

হবে। এরপর এই গড়টি ছাপিয়ে আগের মতই থেমে যাবে। সুতরাং যতগুলি সংখ্যার গড় বের করারই প্রয়োজন হোক না কেন, প্রবাহ চিত্রে কোনো নির্দেশেরই পরিবর্তনের দরকার নেই।

উদাহরণ 5.

দুটি সংখ্যার গরিষ্ঠ সাধারণ গুণনীয়ক বের করতে হবে।

সমাধান ঃ

প্রথমে এই সমস্যাটির সমাধান একটি সাধারণ ভাষায় লিখলে কিরকম হবে তা দেখানো হচ্ছে।

1 সংখ্যক নির্দেশ ঃ সংখ্যা দুটি পড়ে যথাক্রমে A এবং B নামের দুটি স্মৃতিকোষে রাখা হবে এবং রাখার সময়ে B-তে বড় সংখ্যাটি রাখতে হবে।

2 সংখ্যক নির্দেশ ঃ B-এর সংখ্যাটিকে A-এর সংখ্যাটি দিয়ে ভাগ করে ভাগশেষটি C নামের স্মৃতিকোষে রাখা হল।

3 সংখ্যক নির্দেশ ঃ যদি C-এর সংখ্যাটি শূন্য হয় তবে 7 সংখ্যক নির্দেশে এবং না হলে 4 সংখ্যক নির্দেশে যাবে।

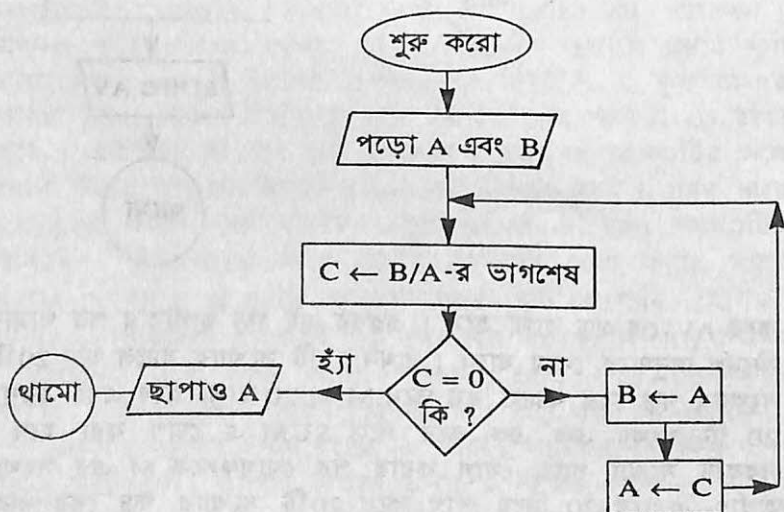
4 সংখ্যক নির্দেশ ঃ A-এর সংখ্যাটি B-তে রাখবে।

5 সংখ্যাক নির্দেশ ঃ C-এর সংখ্যাটি A-তে রাখবে।

6 সংখ্যক নির্দেশ ঃ 2 সংখ্যক নির্দেশে আবার ফিরে যাবে।

7 সংখ্যক নির্দেশ ঃ A-এর সংখ্যাটি ছাপাবে।

8 সংখ্যক নির্দেশ ঃ থেমে যাবে।

এবারে উপরের সমস্যাটির প্রবাহ চিত্র দেওয়া হল।

এখন যে কোনো দুটি সংখ্যার সাহায্যে প্রবাহ চিত্রটি কিভাবে কাজ করছে দেখানো যেতে পারে। মনে করা যাক, দ্বিতীয় নকশাটি অনুসারে B-তে 35 এবং A-তে 14 রাখা হল। এরপর 35-কে 14 দিয়ে ভাগ করার পর C-তে ভাগশেষটি 7 রাখা হবে। এই ভাগশেষ বিভিন্ন উপায় বের করা সম্ভব। কমপিউটারের বিভিন্ন উচ্চ পর্যায়ের ভাষায় এজন্য এক এক রকমের নির্দেশ দেওয়ার ব্যবস্থা আছে। এখানে ধরে নেওয়া হচ্ছে, ওইরকম কোনো নির্দেশের সাহায্যে C-তে ভাগশেষ রাখা যাবে। এরপরের নকশাটির নির্দেশ করার সময়ে C-তে শূন্য নেই বলে 'না'-এর দিকের তীর চিহ্ন অনুসরণ করে B-তে 14 এবং A-তে 7 রাখা হবে। এরপর আবার 14-কে 7 দিয়ে ভাগ করে ভাগশেষ শূন্য হবে এবং সেই ভাগশেষ C-তে থাকবে। এরপরের নির্দেশ অনুযায়ী C-তে শূন্য থাকায় 'হ্যাঁ'-এর দিকের তীরচিহ্ন অনুসরণ করে A-এর সংখ্যাটি অর্থাৎ 7 ছাপিয়ে এরপরের নির্দেশ অনুযায়ী থেমে যাবে। 7-ই সংখ্যা দুটির গরিষ্ঠ সাধারণ গুণনীয়ক। এই প্রবাহ চিত্রে লক্ষ্য করার বিষয়, বারবার কতগুলি নির্দেশ করার জন্য যে নির্দেশ তিনটির কথা আগের উদাহরণে উল্লেখ করা হয়েছিল এখানে তাদের ব্যবহার ছাড়াই কয়েকটি নির্দেশ বারবার করা সম্ভব হয়েছে।

এতক্ষণ যেসব সমস্যার উল্লেখ করা হয়েছে সেগুলির সমাধানের সময়ে কিছু নির্দিষ্ট সংখ্যক নির্দেশ ঠিক কতবার করতে হবে তা আগে থেকেই জানা রয়েছে। কাজেই যতবার করার কথা তা হয়ে গেলেই থামানোর নির্দেশটি পালন করবে। আবার এমন অনেক সমস্যা আছে যেখানে কিছু নির্দিষ্ট সংখ্যক নির্দেশ বারবার করার দরকার হলেও সঠিক কতবার করতে হবে তা আগে থেকে জানা সম্ভব নয়। এক্ষেত্রে থামানোর জন্য অন্য ব্যবস্থা নিতে হবে। এবারের উদাহরণে এই ধরনের একটি সমস্যা এবং তার সমাধান দেখানো হচ্ছে।

উদাহরণ 6.

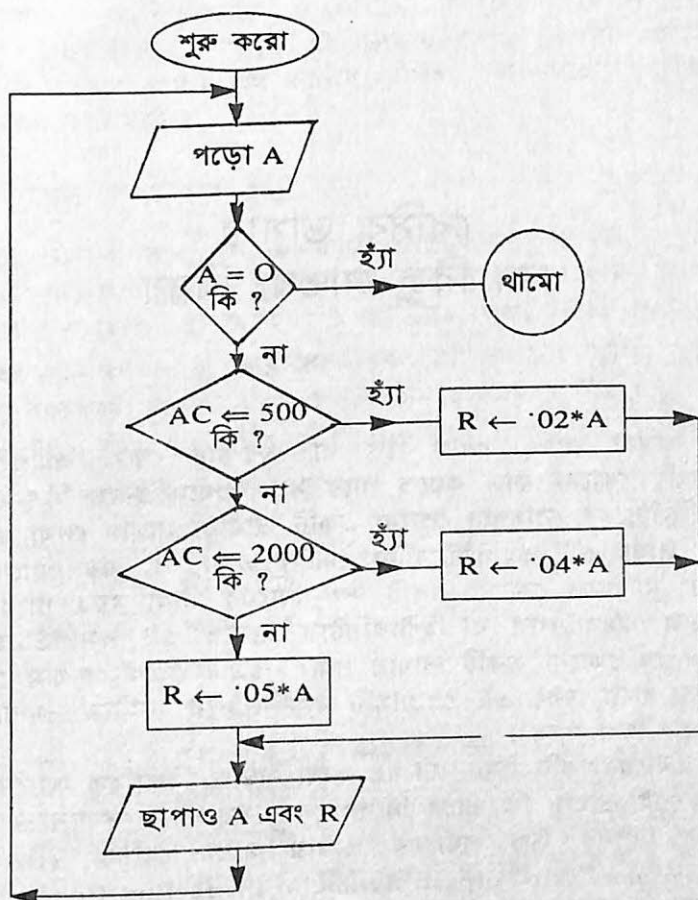
একটি সমবায়িকাতে ক্রেতাদের জিনিস কেনার উপর নির্ভর করে ওই জিনিসের ক্রয় মূল্যের উপর ছাড় দেওয়া হয়। একজন ক্রেতা কত ছাড় পাবেন তা নীচের নীতি অনুসরণ করে বের করা যাবে।

ক্রয়মূল্যের পরিমাণ (টাকায়)	ছাড়ের পরিমাণ
500 পর্যন্ত	2%
500-র বেশি কিন্তু 2000-এর বেশি নয়	4%
2000-এর বেশি	5%

সমাধান ঃ

একজন ক্রেতা যখন কোনো কিছু কিনবেন তখন মোট ক্রয়মূল্যের পরিমাণ কমপিউটারকে জানিয়ে দেওয়া হবে। কমপিউটারের সাহায্যে উপরের নীতি অনুসরণ করে ছাড়ের পরিমাণ তখন ছাপানো হবে। এরপর আবার পরের ক্রেতা। সেখানে মোট ক্রয়মূল্যের পরিমাণ জেনে তাঁর ছাড়ের পরিমাণ আবার জানানো হবে। এইভাবে চলবে একের পর এক ক্রেতার হিসেব। কিন্তু এভাবে এক একজন ক্রেতার হিসেব পাওয়া গেলেও এক্ষেত্রে সেদিন মোট কতজন ক্রেতা আসবেন আগে থেকে তা জানা কখনোই সম্ভব নয়। কাজেই কমপিউটারে অ্যালগরিদম্‌টিকে থামানোর জন্য এখানে অন্য ব্যবস্থা নিতে হবে। সেই ব্যবস্থাটা কি ? এখানে একজনের মোট ক্রয়মূল্য পড়ে নিয়ে ওই মূল্যের জন্য যা ছাড় পাওয়া সম্ভব তা বের করে ছাপানো হবে। তারপরে যে সব জিনিস কিনেছেন পরের ক্রেতা তার মোট ক্রয়মূল্য পড়া হবে। এইভাবে একের পর এক পড়া চলবে এবং ছাড় বের করে ছাপানো হবে। আর যখন কোনো ক্রেতা থাকবে না অর্থাৎ ক্রয়মূল্য যখন শূন্য পড়া হবে তখনই থামার প্রয়োজন হবে। কাজেই একজনের জিনিসের মোট ক্রয়মূল্য জানার পরই তা শূন্য কিনা পরীক্ষা করে দেখে নেওয়া প্রয়োজন। শূন্য হলে থামবে, না হলে ওই মূল্যের উপর ছাড় বের করে ছাপিয়ে আবার আর একজনের মোট ক্রয় করা জিনিসের মূল্য পড়া হবে।

এবারে প্রবাহ চিত্রের সাহায্যে এই সমস্যাটির সমাধান দেখানো হচ্ছে।

এখানে প্রথমে একজন ক্রেতার মোট ক্রয়মূল্যের পরিমাণ পড়ে A নামের স্মৃতিকোষে রাখা হল। এরপরের নকশাটিতে একটি সিদ্ধান্ত নেওয়া হচ্ছে। যদি A-তে শূন্য থাকে অর্থাৎ আর কোনো ক্রেতা না থাকলে হ্যাঁ-এর দিকের তীরচিহ্ন অনুসরণ করে থামার নির্দেশটি পালন করবে। অপরদিকে A শূন্য না হলে প্রথমে একটি সিদ্ধান্ত নেওয়া হচ্ছে A-তে যা আছে তা 500-এর থেকে কম বা সমান না বেশি। যদি কম বা সমান হয় তাহলে 2% হিসেবে ছাড় বের করে ছাপানো হবে। বেশি হলে আবার সিদ্ধান্ত নেওয়া হচ্ছে A-তে যা আছে তা 2000-এর কম বা সমান না বেশি। কম বা সমান হলে 4% হিসেবে ছাড়, কিন্তু বেশি হলে 5% হিসেবে ছাড় দেওয়া হবে। ছাড় বের করার পর প্রত্যেক ক্ষেত্রেই তা ছাপানোর ব্যবস্থা করা হয়েছে। ছাপানোর পরে আবার পরের ক্রেতার মোট ক্রয়মূল্য পড়া হচ্ছে। এইভাবে একের পর এক ক্রেতার ছাড়ের পরিমাণ বের করা হবে। এইভাবে চলতে চলতে এক সময়ে যখন আর কোনো ক্রেতা থাকবে না তখন মোট ক্রয়মূল্য শূন্য দেওয়া হবে এবং এই শূন্য পাওয়া

মাত্রই আর যে কোনো ক্রেতা নেই কমপিউটার তা বুঝতে পারবে। কাজেই তখন সে থেমে যাবে।

বেসিক ভাষার মূল কিছু জ্ঞাতব্য বিষয়

আমরা জানি, প্রবাহ চিত্র বা ফ্লো-চার্ট থেকে কমপিউটার সরাসরি কোনো কাজ করতে পারে না। প্রথমে প্রবাহ চিত্র থেকে কমপিউটারের বোধগম্য কোনো একটি ভাষায় প্রোগ্রাম লেখা হয়। এই ভাষা ওই কমপিউটারটির মেশিন ভাষা, বা ওর অ্যাসেম্বলি ভাষা না হলে কোনো একটি উচ্চ-পর্যায়ের ভাষা হতে পারে যে ভাষার কমপাইলার বা ইন্টারপ্রিটার রয়েছে ওই কমপিউটারে। এখন যে কোনো একটি ভাষায় প্রবাহ চিত্রটি থেকে প্রোগ্রাম লেখা যেতে পারে এবং এই প্রোগ্রামটি কমপিউটারে চালিয়ে সমস্যাটির সমাধান করা সম্ভব।

এবারে একটি উচ্চ পর্যায়ের ভাষা সম্বন্ধে বিস্তারিত আলোচনা করে ওই ভাষায় কি ভাবে নির্দেশ লেখা হয় তাও দেখানো হবে। নানা ধরনের উচ্চ পর্যায়ের ভাষার মধ্যে 'বেসিক' (BASIC –Beginners All Purpose Symbolic Instruction Code) ভাষা সম্বন্ধে এখানে আলোচনা করা হবে। এই ভাষাটি শেখা তুলনা-মূলকভাবে সহজ এবং পার্সোনাল কমপিউটারে এই ভাষার প্রচলনই বেশি। অবশ্য পার্সোনাল কমপিউটার ছাড়াও অন্যান্য কমপি-উটারেও এই ভাষার কমপাইলার এবং ইন্টারপ্রিটার পাওয়া যায়।

ডার্টমাউথ কলেজের ড. জন কেমেনি এবং টম্যাস ই-কুর্জ 1964 সালের 1 মে এই ভাষাটি সৃষ্টি করেন। এরপর বিভিন্ন কমপিউটার তৈরির সংস্থা তাঁদের সুবিধেমত এই ভাষাটিকে অল্পবিস্তর পরিবর্তন করে নিজেদের যন্ত্রে ব্যবহার করতে আরম্ভ করেন। এখানে যে ধরনের বেসিক ভাষা আলোচনা করা হবে তা যে কোনো পার্সোনাল কমপিউটারেই চালানো সম্ভব। এই বেসিক ভাষাটি "এম-বেসিক' (MBASIC) নামে প্রচলিত। মাইক্রো সফ্‌ট্ কোম্পানি এই ভাষাটির প্রচলন করেন। তবে এ ভাষার ইন্টারপ্রিটার বিভিন্ন নামে পাওয়া যায়, যেমন, BASICA, GW-BASIC। যে-সব সংস্থার

পার্সোনাল কমপিউটারে BASICA কিংবা GW-BASIC ইন্টারপ্রিটার থাকবে সেখানে এই ভাষায় লেখা কোনো প্রোগ্রাম চালানো যাবে। এখানে এই ধরনের বেসিক ভাষা সম্বন্ধে বিস্তারিত আলোচনা করা হবে।

বেসিকের বর্ণমালা ঃ

যে কোনো ভাষা শেখার সময়ে সেই ভাষার বর্ণমালার সঙ্গে সর্বাগ্রে পরিচিত হওয়া প্রয়োজন। এই বেসিক ভাষায় যে-সব বর্ণমালার ব্যবহার করা যেতে পারে তা নীচে দেওয়া হলঃ

1. ইংরেজি বর্ণমালার সমস্ত অক্ষর – বড় হাতের এবং ছোট হাতের দু-রকমেরই, অর্থাৎ A থেকে Z আবার a থেকে z পর্যন্ত।
2. দশটি দশমিক অঙ্ক – 0, 1, 2, 3, 4, 5, 6, 7, 8 এবং 9।
3. নীচের কয়েকটি বিশেষ চিহ্ন

চিহ্ন	নাম
	ফাঁকা চিহ্ন
=	সমান চিহ্ন
+	যোগ চিহ্ন
–	বিয়োগ চিহ্ন
*	তারকার চিহ্ন বা গুণ চিহ্ন
/	ভাগ চিহ্ন
^ বা ↑	সূচক চিহ্ন
(	বামদিকের প্রথম বন্ধনী বা লঘু বন্ধনী
)	ডানদিকের প্রথম বন্ধনী বা লঘু বন্ধনী
%	শতকরা চিহ্ন
#	সংখ্যা চিহ্ন
!	বিস্ময় সূচক চিহ্ন
$	ডলার চিহ্ন
[	বামদিকের তৃতীয় বন্ধনী বা গুরু বন্ধনী
]	ডানদিকের তৃতীয় বন্ধনী বা গুরু বন্ধনী
,	কমা চিহ্ন
.	দশমিক চিহ্ন
;	সেমিকোলন চিহ্ন
:	কোলন চিহ্ন

চিহ্ন	অর্থ
&	'অ্যামফারস্যাণ্ড'
?	জিজ্ঞাসা চিহ্ন
<	ক্ষুদ্রতর চিহ্ন
>	বৃহত্তর চিহ্ন
\	উল্টোভাবে ভাগ চিহ্ন। দুটি সংখ্যার মধ্যে এই চিহ্ন ব্যবহার করলে ভাগফল একটি অখণ্ড সংখ্যা পাওয়া যায়।
@	হার অনুসারে
'	সম্বন্ধপদ চিহ্ন বা উদ্ধৃতি চিহ্ন
_	'আণ্ডার স্কোর'

বেসিকে ধ্রুবক ঃ

কমপিউটারে বিভিন্ন সমস্যা সমাধান করার বেলায় অনেক সময়ে যেমন সংখ্যার প্রয়োজন আছে, তেমনি আবার মানুষের নাম-ঠিকানা, ফলাফলের শিরোনামের মত খবরাখবর দরকার হয়। কাজেই বেসিক দু-ধরনের ধ্রুবকেরই ব্যবস্থা যুক্ত। এক ধরনের ধ্রুবকে বেসিকের যে কোনো অক্ষর বা চিহ্ন থাকা সম্ভব। কারো নাম, ঠিকানা কিংবা ফলাফলের শিরোনামের জন্যে এইসব ধ্রুবকের প্রয়োজন। এদের বর্ণ ও সংখ্যামালা সূচক ধ্রুবক বা সারি ধ্রুবক বলা যেতে পারে। এখানে অক্ষরগুলি সারিবদ্ধ অবস্থায় পরপর থাকে বলেই এদের নামকরণ সারি ধ্রুবক। অন্য আর একরকম ধ্রুবক কেবলমাত্র সংখ্যা বোঝানোর জন্য ব্যবহৃত হওয়ায় এরা সংখ্যা সূচক ধ্রুবক নামে অভিহিত।

বর্ণ ও সংখ্যামালা সূচক বা সারি ধ্রুবকে বেসিক বর্ণমালার সব অক্ষরই, এমনকি সংখ্যাও থাকতে পারে। এই ধরনের ধ্রুবকে একসঙ্গে সারি দিয়ে 255টি পর্যন্ত অক্ষর বা চিহ্ন থাকা সম্ভব। এই ধ্রুবক বোঝানোর জন্য অক্ষরগুলির শুরুতে দুটি এবং শেষে দুটি উদ্ধৃতি চিহ্ন থাকে। যেমন,

1. "HELLO"— এটি একটি সারি ধ্রুবক এবং এখানে মোট 5টি অক্ষর আছে।
2. "THE FREQUENCY TABLE"— এই ধ্রুবকটি মোট 19টি অক্ষর যুক্ত। এই 19টির মধ্যে E এবং F-এর মধ্যেকার এবং Y ও T-এর ভেতরের ফাঁকা জায়গা দুটিকেও ধরা হয়েছে। দুদিকের উদ্ধৃতি চিহ্নর মধ্যেকার সবকটি অক্ষরই এখানে ধরা আছে। কিন্তু দু দিকের উদ্ধৃতি

চিহ্নগুলি এই গণনার মধ্যে নিয়ে আসা হয় না।

3. "25.32"— দশমিক চিহ্নটি ধরে এখানে মোট 5টি অক্ষর বা চিহ্ন আছে।

4. "A+B"— এখানে মোট 3টি (যোগ চিহ্নটি নিয়ে) অক্ষর বা চিহ্ন আছে।

5. "$AMOUNT"— এখানে মোট 7টি অক্ষর বা চিহ্ন আছে।

সংখ্যা সূচক ধ্রুবক আবার কয়েক রকমের হয়, যেমন, অখণ্ড সংখ্যা, সূচকবিহীন সংখ্যা এবং সূচকযুক্ত সংখ্যা।

i) অখণ্ড সংখ্যা – এই ধরনের ধ্রুবকে –32768 এবং +32767 এর মধ্যে যে কোনো একটি পূর্ণ সংখ্যা হতে পারে। সাধারণত একটি পার্সোনাল কমপিউটার 16 বিট যুক্ত, সুতরাং এই কম-পিউটারে সবচেয়ে বড় পূর্ণ সংখ্যা হবে $2^{15}-1$ অর্থাৎ 32767 এবং সবেচেয়ে ছোট পূর্ণ সংখ্যা –32768। এই রকম ধ্রুবকে কোনো দশমিক চিহ্ন থাকে না, যেমন, –5, 513, –7123, 62।

ii) সূচকবিহীন সংখ্যা – এই ধরনের ধ্রুবকে দশমিক চিহ্ন ব্যবহার করা হয় এবং এরা ধনাত্মক বা ঋনাত্মক দু রকমেরই হতে পারে, যেমন, 10.3, –30.67, 567.83।

iii) সূচক সংখ্যা – এই ধরনের ধ্রুবকও ধনাত্মক বা ঋনাত্মক দু-ধরনেরই হওয়া সম্ভব। এই রকম সংখ্যা 10-এর সূচক হিসেবে লেখা হয় এবং E-অক্ষরটির সাহায্যে তা বোঝানো হয়, যেমন, –23E+6 অর্থাৎ $-23 \times 10^{6} = -23000000$, 3.457E–7 অর্থাৎ $3.457 \times 10^{-7} = .0000003457$। এই ধরনের সংখ্যা 10^{-38} এবং 10^{38}-এর মধ্যে হয়। এর কারণ সূচক বোঝানোর জন্য যে কটি বিট নির্দিষ্ট আছে তাতে এই সংখ্যা দুটির বাইরে আরও কোনো ছোট বা বড় সংখ্যা হওয়া সম্ভব নয়। এই সংখ্যাগুলিতে দশমিক চিহ্ন থাকা বা না থাকা উভয়ই সম্ভব।

সূচক-সংখ্যা আবার দুভাবে বোঝানো যায়। আগেই বলা হয়েছে, যে কোনো একটি

দশমিক ভগ্নাংশ কমপিউটারে সঠিকভাবে সঞ্চয় করা সম্ভব নাও হতে পারে। অনেক সময়েই দ্বি-নিধানী বা বাইনারি সংখ্যায় পরিবর্তনের পর ভগ্নাংশটির কাছাকাছি একটি আসন্ন সংখ্যা পাওয়া যায়। এর কারণ, ভগ্নাংশ রাখার জন্য যে কটি বিট নির্দিষ্ট থাকে, তা অনেক সময়েই পর্যাপ্ত নয়। ভগ্নাংশ রাখার জন্য আরও বেশি সংখ্যক বিটের ব্যবস্থা করা সম্ভব হলে দ্বি-নিধানী সংখ্যায় পরিবর্তিত সংখ্যাটি আগের চেয়ে ভগ্নাংশটির অনেকটাই কাছাকাছি হবে। অর্থাৎ এই নতুন সংখ্যাটি আরও সঠিক হবে, বলা চলে। এই বেশি সংখ্যক বিট নির্দিষ্ট করার জন্য সংখ্যাটি লেখার সময় E-এর বদলে D লিখতে হয়। D লেখা সংখ্যাগুলির জন্য বেসিকের ইন্টারপ্রিটার বা কমপাইলার E লেখা সংখ্যার থেকে সংখ্যায় বেশি বিট নির্দিষ্ট করে। দশমিক অখণ্ড সংখ্যা ছাড়া অন্যান্য সব সংখ্যাকেও সাধারণত দুভাবে দেখা যেতে পারে, যেমন, একক-দৈর্ঘ্য এবং দ্বি-দৈর্ঘ্য সংখ্যা। একক-দৈর্ঘ্যের সংখ্যা হয় সূচকবিহীন না হয় E সূচকযুক্ত। এরা সাধারণত

1) 7 অথবা তার চেয়ে কম সংখ্যক অঙ্কের হতে পারে, যেমন, 56.2, −3281.6, 4678230।

2) E যুক্ত একটি সংখ্যা হতে পারে, যেমন, −1.23E−06, 23.46E05।

3) অনেক সময়ে 7-এর চেয়ে কম অঙ্কের ক্ষেত্রে সংখ্যাটির শেষে একটি বিস্ময়-সূচক চিহ্ন (!) ব্যবহার করা হয়ে থাকে, যেমন, 13.2!। এর ফলে এই ধরনের সংখ্যা 6 অঙ্ক পর্যন্ত নিখুঁত পাওয়া যাবে।

দ্বি-দৈর্ঘ্য সংখ্যা একক-দৈর্ঘ্যের সংখ্যাগুলির চেয়ে দ্বিগুণ নিখুঁত হয় বলে এইরকম নামকরণ। এই ধরনের সংখ্যা

1) 8 অথবা তার চেয়ে বেশি কিন্তু 17-এর

চেয়ে বেশি সংখ্যক অঙ্কযুক্ত এরা নয়, যেমন, 123456789, – 98765.23412।

2) D যুক্ত একটি সংখ্যা হওয়াও সম্ভব, যেমন, –1.23D–06।

এবং

3) শেষ চিহ্নটি একটি সংখ্যা চিহ্ন (#), যেমন, 13.2 #। কোনো 8 অঙ্কের কম দশমিক ভগ্নাংশকে দ্বি-দৈর্ঘ্য নিখুঁত করতে হলে শেষ অঙ্কর পরে একটি সংখ্যা চিহ্ন ব্যবহার করা হয়।

এই সব দ্বি-দৈর্ঘ্য সংখ্যায় 17 অঙ্ক রাখা হলেও প্রথম 16টি অঙ্ক ছাপা হয় যেমন একক-দৈর্ঘ্যের ক্ষেত্রে 7টি অঙ্ক রাখা হলেও 6টি অঙ্ক পর্যন্ত নিঁখুত পাওয়া যায়।

উপরে যে-সব সংখ্যা সম্বন্ধে আলোচনা করা হল সেসব ছাড়াও আরও দু-ধরনের সংখ্যার প্রচলন আছে। যেমন,

iv) হেক্স সংখ্যা – এই ধরনের সংখ্যায় 10-এর ভিত্তির বদলে 16 ব্যবহার করা হয় এবং এই রকম সংখ্যা যাতে বোঝা যায় সে জন্য সংখ্যার আগে & H এই দুটি চিহ্ন লেখা হয়ে থাকে। দৃষ্টান্তস্বরূপ, &H79 এই হেক্স সংখ্যাটির সমান দশমিক সংখ্যা হবে

$7 \times 16^1 + 9 \times 16^0 = 112 + 9 = 121$

আবার &H15C $= 1 \times 16^2 + 5 \times 16^1 + C \times 16^0$

$= 256 + 80 + 12 = 348$

অপরদিকে দশমিক পূর্ণ সংখ্যার সমান হেক্স সংখ্যা বের করতে হলে ক্রমান্বয়ে 16 দিয়ে ভাগ করে যেতে হবে। এই সংখ্যা-পদ্ধতিতে দশমিক সংখ্যার দশটি অঙ্ক 0 থেকে 9 ছাড়াও 10, 11, 12, 13, 14 এবং 15 বোঝানোর জন্য যথাক্রমে A, B, C, D, E এবং F ব্যবহার করা হয়। এই সংখ্যা-পদ্ধতির ভিত্তি 16 বলে 0 থেকে 15-এই 16টি অঙ্কের জন্য 16টি ভিন্ন ভিন্ন চিহ্নর ব্যবস্থা রাখতে হয়েছে। এই চিহ্নগুলি হল 0, 1, 2, 3, 4, 5,

6, 7, 8, 9, এবং A, B, C, D, E, F।

v) অক্টাল সংখ্যা – এই ধরনের সংখ্যা-পদ্ধতির ভিত্তি 8 এবং এই সংখ্যা বোঝানোর জন্য সংখ্যার আগে &0 বা শুধুমাত্র & লেখা হয়। সংখ্যার ভিত্তি 8 হওয়ায় 8টি অঙ্কের জন্য 8টি চিহ্ন হল 0, 1, 2, 3, 4, 5, 6, এবং 7। এক্ষেত্রে দশমিক কোনো পূর্ণ-সংখ্যার সমান অক্টাল সংখ্যা বের করতে হলে ভাগফল 8-এর কম না হওয়া পর্যন্ত ক্রমান্বয়ে 8 দিয়ে ভাগ করে যেতে হবে। কাজেই 91_{10} এর সমান অক্টাল সংখ্যা হবে

```
8 | 91
  |----
8 | 11   3
  |----
     1   3
```

অর্থাৎ 133_8। আবার 53_8 এই অক্টাল সংখ্যার সমান দশমিক সংখ্যা হবে $5 \times 8^1 + 3 \times 8^0 = 40 + 3 = 43_{10}$।

চলরাশি ঃ

বেসিক ভাষায় যে-সব বিভিন্ন ধরনের ধ্রুবকের ব্যবহার দেখা যায়, এতক্ষণ পর্যন্ত সে সম্পর্কে আলোচনা করা হল। কোনো একটি সমস্যার সমাধানে যেমন ধ্রুবকের দরকার তেমনি আবার চল-রাশিরও প্রয়োজন। ধ্রুবকের মত এই চলরাশিও বিভিন্ন ধরনের। এক এক ধরনের চলরাশিতে এক এক রকমের ধ্রুবক রাখা হয়। চলরাশি এবং ধ্রুবকের মধ্যে পার্থক্য, একটি প্রোগ্রাম চলাকালীন একটি চলরাশির মান এক এক সময়ে এক এক রকম হতে পারে, কিন্তু ধ্রুবকের মান অপরিবর্তনীয়। উদাহরণস্বরূপ বলা যায়, চলরাশির মান শুরুতে 0 হতে পারে। তারপর ক্রমান্বয়ে 1 যোগ করায় এই চলরাশির মান 1, 2, 3, ইত্যাদি হওয়া সম্ভব।

একটি চলরাশির নামকরণে যে-সব অক্ষর ব্যবহার করা যায় সেগুলি উল্লেখ করা যাক। প্রথমে তা হলঃ

1. ইংরেজি বর্ণমালার সব অক্ষর অর্থাৎ A থেকে Z—বড় অক্ষর বা ছোট অক্ষর। দু-ধরনের অক্ষরকেই একই ধরা হয়।
2. দশমিক সংখ্যা-পদ্ধতির দশটি অঙ্ক অর্থাৎ 0 থেকে 9।
3. দশমিক চিহ্ন (.)।

এই নামকরণের সময় নীচের নিয়মগুলি মনে রাখা প্রয়োজন ।

1. ইংরেজি বর্ণমালার যে কোনো একটি অক্ষর নিয়ে নামকরণ শুরু করতে হবে ।
2. নামের মোট অক্ষর সংখ্যা 40-এর বেশি হলে প্রথম 40টি অক্ষরকে নাম হিসেবে ধরে নেওয়া হবে ।
3. বেসিক ভাষার নির্দেশ বোঝানোর জন্য যে-সব নাম ব্যবহার করা হয় সেগুলি কোনো চলরাশির নাম হিসেবে ব্যবহার করা যাবে না ।
4. এই নাম কখনোই FN দিয়ে শুরু হবে না ।

এবারে নামের কয়েকটি উদাহরণ দেওয়া যাক ।

A— এই নামে ইংরেজি বর্ণমালার শুধুমাত্র A অক্ষরটি ব্যবহার করা হয়েছে । এই নামে কেবলমাত্র একটি অক্ষর আছে ।

B12— এখানে ইংরেজি বর্ণমালার B অক্ষর দিয়ে শুরু এবং পরের দুটি দশমিক সংখ্যার অঙ্ক ব্যবহার করা হয়েছে । এই নামের মোট অক্ষর সংখ্যা 3 ।

ABCD— এই নামটি ইংরেজি বর্ণমালার A অক্ষরটি দিয়ে শুরু এবং পরের তিনটি অক্ষরও ইংরেজি বর্ণমালা থেকে নেওয়া হয়েছে ।

এবারে কয়েকটি অশুদ্ধ নামের নমুনা দেওয়া হচ্ছে।

A+B— এখানে যদিও ইংরেজি বর্ণমালার A অক্ষরটি দিয়ে শুরু হয়েছে কিন্তু দ্বিতীয় চিহ্নটি (যোগ চিহ্ন) নামে ব্যবহার করার নিয়ম নেই । কাজেই এটি একটি চলরাশির নাম হওয়া সম্ভব নয় ।

12C— এক্ষেত্রে প্রথম অক্ষরটি ইংরেজি বর্ণমালার কোনো একটি অক্ষর না হয়ে দশমিক সংখ্যার একটি অঙ্ক ব্যবহার করা হয়েছে । সুতরাং এই নামটিও শুদ্ধ নয় ।

DATA— এটি যদিও ইংরেজি বর্ণমালার D দিয়ে শুরু এবং নামের চারটি অক্ষরই ইংরেজি বর্ণমালার কিন্তু এটি বেসিক ভাষার নির্দেশ বোঝানোর জন্য যে-সব নাম ব্যবহার করা হয় তার একটি । কাজেই এই নামটিও চলরাশির নাম হিসেবে ব্যবহার করা যাবে না ।

C D— এখানে C এবং D-এর মধ্যে একটি ফাঁকা জায়গা আছে । নামে কোন ফাঁকা জায়গা থাকে না । কাজেই এটিও চলরাশির নাম হবে না ।

ধ্রুবকের মত চলরাশিও দু রকমের হয়, সারি চলরাশি এবং সংখ্যা চলরাশি।

যে-সব স্মৃতিকোষে বর্ণ ও সংখ্যামালা-সূচক ধ্রুবক রাখা হবে সেইসব নামের শেষ অক্ষরটি একটি ডলার চিহ্ন ($) হয়, যেমন, A$, ABC$, M12$ । অর্থাৎ নামের শেষ অক্ষরটি দেখেই বোঝা যাবে এখানে কি ধরনের ধ্রুবক রাখা হবে। এই ধরনের চলরাশির জন্য স্মৃতিতে কটি বাইট নির্দিষ্ট থাকবে ? এটি অবশ্য সারি ধ্রুবকে কটি অক্ষর আছে তার উপরে নির্ভর করবে। তাহলে A$ নামের চলরাশিতে যদি 'A' রাখা হয় তাহলে দুটি উক্তি চিহ্নর মধ্যে একটি মাত্র অক্ষর A থাকায় A$ নামের স্মৃতিকোষেও মোট একটি অক্ষর A থাকবে। আবার A$-এ 'BASIC PAY' রাখা হলে, দুটি উক্তি চিহ্নর মধ্যে মোট 9টি অক্ষর থাকায় (BASIC এবং PAY-এই শব্দ দুটির মধ্যের ফাঁকা জায়গাটি ধরে) এবারে এই অক্ষরগুলির জন্য 9টি বাইটের প্রয়োজন হবে।

সংখ্যা চলরাশি আবার তিন রকমের। এক এক ধরনের চলরাশির নাম এক এক রকমের সংখ্যা সূচক ধ্রুবক রাখার জন্য ব্যবহৃত হয়।

কোনো চলরাশির নামের শেষ অক্ষরটি শতকরা চিহ্ন (%) হলে এই চলরাশিতে কেবলমাত্র দশমিক অখণ্ড সংখ্যা রাখা যাবে। যেমন A%, A12% । এই ধরনের চলরাশির মান রাখার জন্য 2টি বাইট অর্থাৎ 16টি বিট নির্দিষ্ট থাকে। কাজেই এই ধরনের চলরাশির মান সব সময়েই –32868 এবং 32767-এর মধ্যে থাকবে।

একক-দৈর্ঘ্য এবং দ্বি-দৈর্ঘ্য দু-ধরনের সংখ্যার জন্য চলরাশির নামের শেষ অক্ষরটিও আবার দু-রকমের হয়। যে-সব চলরাশির নামের শেষ অক্ষরটিতে কোনো বিশেষ চিহ্ন ব্যবহার করা হয় না অথবা বিস্ময়-সূচক চিহ্নটি (!) থাকে সে-সব চলরাশির মান একক-দৈর্ঘ্য সংখ্যা হতে পারে। যেমন, B, AB, B12!, ROOT1 । এই ধরনের চলরাশির জন্য 4টি বাইট নির্দিষ্ট থাকে। চতুর্থ চলরাশির নামে মোট পাঁচটি অক্ষর আছে শেষের দশমিক অঙ্ক 1 ধরে।

যে সকল সমস্যার দশমিক ভগ্নাংশগুলি খুব নিখুঁত হওয়ার প্রয়োজন আছে সেখানে চলরাশির নামের শেষ অক্ষরটিতে একটি সংখ্যা চিহ্ন (#) ব্যবহার করা হয়, যেমন, R2#, P#, ABC# । এই ধরনের চলরাশির জন্য একক-দৈর্ঘ্যের চেয়ে দ্বিগুণ সংখ্যক বাইট নির্দিষ্ট থাকে।

এখানে উল্লেখ করা দরকার, নামে যে-সব অক্ষর বা চিহ্ন ব্যবহার করার কথা আগে বলা হয়েছে সেখানে %, !, #-এর মত চিহ্নের উল্লেখ ছিল না। কারণ এইসব চিহ্ন নামের অঙ্গ নয়, এদের

ব্যবহার কেবলমাত্র চলরাশির ধরন বোঝানোর জন্য।

আলোচনাতে চলরাশির নামকরণের যে পদ্ধতি বর্ণনা করা হয়েছে তা ব্যবহার না করেও বেসিক ভাষার কিছু নির্দেশের সাহায্যে এই নামকরণ করা সম্ভব। এই নির্দেশগুলি সম্বন্ধে পরে আলোচনা করা হবে।

নির্দেশ লেখার নিয়মাবলী ঃ

একটি সমস্যা সমাধানের জন্য বেসিক ভাষায় পরপর কিছু সংখ্যক নির্দেশ দিয়ে একটি প্রোগ্রাম তৈরি করা হয়। প্রবাহ চিত্রে যেমন বিভিন্ন নকশার সাহায্যে বিভিন্ন ধরনের নির্দেশ বোঝানো হয়, বেসিকেও সেরকম ভিন্ন ভিন্ন নির্দেশের সাহায্যে ভিন্ন ভিন্ন কাজের কথা বলা হয়ে থাকে। এরপর বিভিন্ন ধরনের নির্দেশের কথা আসবে। তবে তার আগে এই ভাষায় নির্দেশ দেওয়ার মৌলিক ব্যাপার সম্বন্ধে জানা প্রয়োজন।

1. প্রত্যেকটি নির্দেশ সাধারণত একটি ভিন্ন লাইনে লেখা হয়। একটি নির্দেশ কয়েকটি অক্ষর এবং চিহ্নের সমষ্টি। একটি লাইনে মোট 255টি পর্যন্ত অক্ষর বা চিহ্ন হওয়া সম্ভব। তবে কমপিউটারের সঙ্গে লাগানো ভিডিও-এর পর্দায় এক লাইনে 80টি অক্ষর ধরানো সম্ভব বলে প্রায় সব নির্দেশই এর মধ্যেই রাখা হয়। কোনো কোনো সময়ে একটি নির্দেশ 80টি অক্ষরের বেশি হলে তা পর্দায় এক লাইনে শুরু হয়ে পরের লাইনেও চলে যায়। আবার অনেক সময়ে এক লাইনে একের বেশিও নির্দেশ লেখা সম্ভব। সেক্ষেত্রে কোলন চিহ্নের (:) সাহায্যে নির্দেশ-গুলিকে আলাদা করা হয়।
2. প্রত্যেক লাইনের শুরুতে একটি পূর্ণ সংখ্যা ব্যবহার করা হয়। এই সংখ্যা 0 থেকে 65529 পর্যন্ত যে কোনো সংখ্যা হতে পারে। এই সংখ্যাকে লাইন-সংখ্যা বলা হয়। এক লাইনে একাধিক নির্দেশ থাকলেও লাইন-সংখ্যা একটিই থাকবে এবং সেটি লাইনের শুরুতেই লেখা হবে। আবার পর্দায় একটি নির্দেশ একাধিক লাইন নিলেও প্রথম লাইনের শুরুতেই কেবলমাত্র লাইন-সংখ্যা থাকবে এবং পর্দায় পরের লাইনে কোনো লাইন-সংখ্যা থাকবে না। প্রত্যেকটি লাইন-সংখ্যা আলাদা হবে। অর্থাৎ কোনো দুটি লাইনে একই লাইন-সংখ্যা ব্যবহার করা যায় না।
3. সাধরণত লাইন-সংখ্যা একের পর এক বেড়ে চলে। তবে লাইন-সংখ্যাগুলি যে ভাবেই থাকুক না কেন অনেক বেসিক ভাষার ইন্টারপ্রিটার প্রোগ্রামটি চলার সময় লাইন-সংখ্যা অনুযায়ী

সাজিয়ে নিয়ে কাজ করে। কাজেই মাঝখানের কোনো নির্দেশ লিখতে ভুলে গেলে পরে তা লেখা যেতে পারে। সেক্ষেত্রে ইন্টারপ্রিটার ঠিকমত সাজিয়ে নেবে।

4. একটি লাইন-সংখ্যা থেকে পরের লাইনসংখ্যার মধ্যে 1-এর তফাত না রেখে সাধারণত 10-এর তফাত রাখা হয়। এর ফলে পরে কোনো সময়ে দুটি লাইন-সংখ্যার মধ্যে নতুন কোনো নির্দেশ লেখার প্রয়োজন হলে পুরনো প্রোগ্রামটির বিশেষ কোনো পরিবর্তনের দরকার হবে না। কেবলমাত্র পুরনো লাইন-সংখ্যা দুটির মধ্যেকার কোনো একটি সংখ্যা ব্যবহার করলেই হবে। উদাহরণস্বরূপ 30 এবং 40 লাইন-সংখ্যা দুটি নেওয়া যাক। এই দুটি লাইন-সংখ্যার মধ্যে কোনো একটি নির্দেশ লিখতে হলে সেই নির্দেশের লাইন-সংখ্যা 35 লিখলে আর কোনো নির্দেশের লাইন-সংখ্যারই পরিবর্তনের প্রয়োজন হবে না। কিন্তু পুরনো লাইন-সংখ্যা 30 এবং 40-এর পরিবর্তে 3 এবং 4 থাকলে নতুন নির্দেশটির লাইন-সংখ্যাটি 4 লেখা দরকার এবং 4 থেকে আরম্ভ করে পরের নির্দেশগুলির লাইন-সংখ্যাগুলি বদলানোর প্রয়োজন হবে।

5. একটি লাইনে লাইন-সংখ্যা লিখে পরের লাইনে চলে গেলেও প্রোগ্রামে কোনো রকম গোলমাল লক্ষ্য করা যাবে না। একটি লাইনে 80টির কম অক্ষর লিখে পরের লাইনে যাওয়ার প্রয়োজন হলে কী বোর্ডের রিটার্ন কী (অর্থাৎ ↵ এই চিহ্নের কী) টিপলেই কাজ হবে।

6. একটি লাইনে লাইন সংখ্যা লেখার পর নির্দেশ লেখার আগে একটি ফাঁকা জায়গা রাখা দরকার। একাধিক ফাঁকা জায়গা থাকলেও ভুল হবে না।

বেসিকে কোনো একটি নির্দেশের জন্য একটি নাম ব্যবহার করা হয়। এই নামটিকে ওই নির্দেশের কী-ওয়ার্ড বা মূল শব্দ হিসেবে ধরা হয়ে থাকে। ওই মূল শব্দটি অনুসারে বেসিক ভাষার ইন্টারপ্রিটার বা কমপাইলার কমপিউটারের যন্ত্রের ভাষায় নির্দেশটিকে অনুবাদ করবে।

এবারে একটি সমস্যা সমাধানের জন্য বেসিকে কি ভাবে নির্দেশ দেওয়া হয় তা দেখা যাক।

মনে করা যাক, দুটি সংখ্যা যোগ করে, যোগফলটি ছাপানোর পর থামতে হবে। বেসিকে উপরের সমস্যাটির সমাধান কেবলমাত্র একটি নির্দেশের সাহায্যেই করা সম্ভব।

10 PRINT 5.2 + 8.3 -এর ক্ষেত্রে কি হবে ?

এখানে যে সংখ্যা দুটি যোগ করা হবে তা ধরা হয়েছে যথাক্রমে

5.2 এবং 8.3 । কোনো কিছু ছাপানোর প্রয়োজন হলে তা PRINT নামের নির্দেশের সাহায্যে করা সম্ভব । এই নির্দেশের আরও একটি সুবিধে, দুটি সংখ্যাকে যোগ করার নির্দেশও এখানে দেওয়া যায় । এখানে উল্লেখ করা প্রয়োজন নির্দেশটির শুরুতেই যে 10 সংখ্যাটি লেখা আছে তা লাইন-সংখ্যা । 10 লাইন-সংখ্যার নির্দেশের পরের লাইনে আর একটি নির্দেশ এই প্রোগ্রামটিকে থামানোর জন্য দেওয়া প্রয়োজন । তা হল 20 END ।

বেসিক ভাষার কয়েকটি সহজ নির্দেশ

অন্যান্য ভাষার মত বেসিক ভাষাতেও এক এক ধরনের কাজের জন্য এক এক রকম নির্দেশ দেওয়া হয়ে থাকে। এবারে এই নির্দেশ-গুলি সম্বন্ধে বিশদ আলোচনা করা যাক।

LET-নির্দেশ ঃ

এই নির্দেশে সমান চিহ্নের বাম পাশে একটি চলরাশির নাম এবং ডান পাশে রাশিমালা বা কোনো সংখ্যা থাকে। অবশ্য রাশিমালা থাকলে তার মান বের করার পরে তবেই তা বাম পাশের নতুন মান হিসেবে ধরা হয়। মনে রাখতে হবে, সমান চিহ্নের বাম পাশে কেবলমাত্র একটি চলরাশির নামই থাকতে পারে। তবে বাম পাশে এক ধরনের চলরাশির নাম এবং ডান পাশে অন্য কোনো এক প্রকার সংখ্যা-সূচক ধ্রুবক থাকা সম্ভব, যেমন, A% = 2.3। এই উদাহরণে বাম পাশের চলরাশির নামের পরে শতকরা চিহ্ন (%) আছে বলে এখানে কেবলমাত্র দশমিক অখণ্ড সংখ্যা রাখা যায়। কিন্তু ডান পাশের সংখ্যাটি একটি একক-দৈর্ঘ্য সংখ্যা। এক্ষেত্রে সংখ্যাটিকে চলরাশিতে রাখার সময় পূর্ণ সংখ্যা হিসেবে তা থাকবে। অর্থাৎ 2.3-এর বদলে কেবলমাত্র 2 আসবে। এখানে একটা কথা উল্লেখ করা দরকার। নির্দেশে LET শব্দটি অনেক সময়ে ব্যবহার না করে উহ্য রাখা হয়ে থাকে। এবারে কিছু উদাহরণ দেওয়া যাক।

উদাহরণ 1.

```
10 LET B% = 53.32
```

এখানে ডান পাশে সূচকবিহীন একটি দশমিক ভগ্নাংশের উল্লেখ করা হয়েছে এবং বাম পাশের চলরাশিটির নামের শেষ অক্ষরটি

শতকরা চিহ্নযুক্ত (%)। সুতরাং এখানে কেবলমাত্র একটি দশমিক অখণ্ড সংখ্যা রাখা যাবে। সুতরাং এই লাইনের নির্দেশ অনুসারে B%-তে 53 থাকবে।

উদাহরণ 2.

20 LET C% = 36.82

এবারে C% নামের চলরাশিতে 36-এর বদলে 37 থাকবে, কারন 36.82 সংখ্যাটিকে অখণ্ড সংখ্যায় পরিবর্তনের সময়ে দশমিক চিহ্নের ডানদিকের প্রথম অঙ্কটি 5 বা তার বেশি কিনা দেখে নেওয়া হয়। অঙ্কটি 5-এর সমান বা তার বেশি হলে দশমিক চিহ্নের আগের অঙ্কের সঙ্গে 1 যোগ করা হয়। এক্ষেত্রে দশমিক চিহ্নের পরে 8 থাকায় 36-এর সঙ্গে 1 যোগ করে 37 পাওয়া যাবে। একে রাউণ্ডিং বলে।

উদাহরণ 3.

30 D = 6#/7

এবারে ডান পাশেই দু ধরনের সংখ্যা-সূচক ধ্রুবক আছে, যেমন, একক-দৈর্ঘ্য এবং দ্বি-দৈর্ঘ্য সংখ্যা।

6 সংখ্যাটির পরে সংখ্যা চিহ্ন (#) থাকায় ভাগ করার সময়ে 16 টি অঙ্ক পর্যন্ত বের করা হবে। সেক্ষেত্রে ভাগফল পাওয়া যাবে .8571428571428571। কিন্তু বাম পাশের চলরাশি একটি একক-দৈর্ঘ্য সংখ্যা চলরাশি, সুতরাং সেখানে মোট 7টি অঙ্ক রাখা হবে এবং অষ্টম অঙ্কটি 5 হওয়ায় সপ্তম অঙ্কের সঙ্গে 1 যোগ করে সংখ্যাটি হবে .8571429। এরপর প্রোগ্রামে যখনই D ব্যবহার করা হবে তার মান ধরা হবে .8571429।

উদাহরণ 4.

40 LET D# = 6#/7

এবারে ডান পাশের ওই 16-টি অঙ্কই D# নামের চলরাশির মান হবে। এর কারণ বাম পাশে একটি দ্বি-দৈর্ঘ্য চলরাশির নাম ব্যবহার করা হয়েছে। অর্থাৎ D# এর মান হবে .8571428571428571।

উদাহরণ 5.

50 LET D# = 6/7

এবারে D# -এর মান হবে .8571428656578064। কেন এমন হল ? ডান পাশের দুটি সংখ্যাই একক-দৈর্ঘ্য সংখ্যা হিসেবে আছে। কাজেই এক্ষেত্রে ভাগফল 7 অঙ্ক পাওয়া যাবে। অর্থাৎ ভাগফল হবে .8571429। কিন্তু বামপাশের চলরাশি একটি দ্বি-দৈর্ঘ্য চলরাশি হওয়ায় এই সাত অঙ্কের সংখ্যাকে দ্বি-দৈর্ঘ্য সংখ্যায় রূপান্তর ঘটানোর পর 16 অঙ্কের নতুন সংখ্যাটি D#-এর মান হবে। আগের

উদাহরণের সঙ্গে এই মানের পার্থক্য হল কেন ? উদাহরণ 4-এ ডান পাশেই দ্বি-দৈর্ঘ্য সংখ্যা থাকায় ভাগফলই একটি 16 অঙ্কবিশিষ্ট সংখ্যা, কিন্তু এই উদাহরণে 7 অঙ্কের ভাগফল তৈরি করার পরে ওই 7 অঙ্ককে একটি 16 অঙ্কের সংখ্যায় পরিবর্তন করা হয়েছে।

উদাহরণ 6.

60 D# = 2.1

এখানে LET ব্যবহার করা হয় নি। অবশ্য প্রথমেই বলা হয়েছে LET শব্দটি অনেক সময়েই ব্যবহার করা হয় না। এক্ষেত্রে D# চলরাশির মান কত হবে ? D#-এর মান হবে 2.099999904632568। এখানে ডান পাশে একটি একক-দৈর্ঘ্য সংখ্যা এবং বাম পাশে একটি দ্বি-দৈর্ঘ্য চলরাশি আছে। কাজেই একক-দৈর্ঘ্য সংখ্যাটিকে দ্বি-দৈর্ঘ্যে পরিবর্তনের পরেই D# নামের স্মৃতিকোষে রাখা হবে। যেমন, উদাহরণ 1-এর বেলাতে ডানপাশের একটি একক-দৈর্ঘ্য সংখ্যাকে একটি অখণ্ড সংখ্যায় পরিবর্তনের পর বামপাশে রাখা হয়েছে।

দুটি সংখ্যা যোগ, বিয়োগ, গুণ বা ভাগ করার সময়তেও নীচের নিয়মগুলি মনে রাখলে কোনো অসুবিধে হবে না।

1. দুটি সংখ্যা একই গোত্রভুক্ত হলে পাটীগণিত করার পর ফলাফল সেই গোত্রেরই থাকবে। অর্থাৎ দুটি একক-দৈর্ঘ্য সংখ্যার বেলায় ফলও একক-দৈর্ঘ্য সংখ্যাই হবে। যেমন উদাহরণ 5-এর বেলায় ভাগফল একক-দৈর্ঘ্য সংখ্যা হওয়ার পরে বাম পাশের চলরাশি অনুযায়ী রূপান্তর ঘটানো হয়েছে।

2. দুটি সংখ্যা যদি ভিন্ন গোত্রের হয়, অর্থাৎ একটি একক-দৈর্ঘ্য সংখ্যা এবং অপরটি দ্বি-দৈর্ঘ্য সংখ্যা হলে সেক্ষেত্রে দ্বি-দৈর্ঘ্য সংখ্যা হিসেবে পাটীগণিতের ফল প্রকাশ করা হবে।

উদাহরণ 7.

10 A = B + C/D – 2.5

উপরের নির্দেশে ডান পাশে কতগুলি চলরাশির নাম, ভগ্নাংশ এবং কয়েকটি পাটীগণিতের চিহ্ন ব্যবহার করা হয়েছে। ডানপাশে পাটীগণিত করার পরে যে মান বেরোবে তা বাম পাশের চলরাশির নতুন মান হবে। এখানে যোগ, বিয়োগ এবং ভাগ এই তিন রকম চিহ্ন ব্যবহার হয়েছে। কিন্তু কোন চিহ্নের কাজ আগে করা হবে ? এক্ষেত্রে সরল সমীকরণ করার নিয়ম মেনে চলা হয়। অর্থাৎ প্রথমে C-এর মানকে D-এর মান দিয়ে ভাগ করার পর যে ফল পাওয়া যাবে তা B-এর মানের সঙ্গে যোগ করে যোগফল থেকে 2.5 বিয়োগ করে বিয়োগফল A-তে রাখা হবে।

ডান পাশে কোন চিহ্নের পরে কোন চিহ্নের কাজ হবে তার নিয়মাবলী নীচে দেওয়া হল ঃ

1. যদি কোনো সূচক-চিহ্ন (^ বা ↑) থাকে তবে তার কাজ হবে সকলের আগে। একের বেশি সূচক-চিহ্ন থাকলে সমান চিহ্নের ডান পাশে প্রথমে যে সূচক-চিহ্নটি পাওয়া যাবে সেটির কাজ আগে করতে হবে, তারপরের কাজ এর পরের চিহ্নটির এবং এইভাবে পরপর কাজ চলবে।

2. সব কটি সূচক-চিহ্ন করার পরে সমান চিহ্নের ডান দিক থেকে গুণ এবং ভাগের মধ্যে যে চিহ্ন আগে থাকবে সেই চিহ্নের কাজই আগে করতে হবে। যদি প্রথমে গুণ, পরে ভাগ এবং আবার একটি গুণ করার চিহ্ন থাকে, তবে ওই ভাবেই কাজ করে যেতে হবে অর্থাৎ প্রথমে প্রথম গুণটি, তারপরে ভাগ এবং শেষে সর্বশেষ গুণ চিহ্নের কাজ করা হবে।

3. সূচক-চিহ্ন, গুণ-চিহ্ন এবং ভাগ-চিহ্নর কাজ হয়ে যাওয়ার পরে আবার সমান চিহ্নের ডান দিক থেকে যোগ এবং বিয়োগ চিহ্নর মধ্যে যে চিহ্ন আগে আসবে, সে চিহ্নের কাজ আগে করতে হবে।

4. লঘু-বন্ধনীর সাহায্যে উপরের নিয়মের পরিবর্তন করা সম্ভব। অর্থাৎ উদাহরণ 7-এ লঘু-বন্ধনী ব্যবহার করে 10 সংখ্যক লাইনের নির্দেশ লেখা যেতে পারে

10 A = (B + C)/D – 2.5

এবারে প্রথমে ভাগ না করে সবচেয়ে আগে B এবং C-এর মান যোগ করে তবেই D-এর মান দিয়ে ভাগ করতে হবে। তারপর ভাগফল থেকে 2.5 বিয়োগ করে যে মান পাওয়া যাবে সেটিই হবে A-এর নতুন মান।

কিন্তু একাধিক লঘু-বন্ধনী থাকলে কি হবে ? যেমন, উদাহরণ 7-এ দুটি লঘু-বন্ধনী ব্যবহার করে লেখা যেতে পারে—

10 A = (B + C) / (D – 2.5)

এবারে কি ভাবে A-এর মান বেরোবে ? এখানে একাধিক লঘু-বন্ধনীর ব্যবহার হয়েছে। এক্ষেত্রে সমান চিহ্নের ডান পাশে প্রথম লঘু-বন্ধনীর কাজ আগে করবে। এরপর হবে দ্বিতীয় লঘু-বন্ধনীর কাজ। এইভাবে পরপর চলতে থাকবে। অর্থাৎ এক্ষেত্রে প্রথমে B এবং C-এর মান যোগ করা হবে। এরপর D-এর মান থেকে 2.5 বিয়োগ করা হবে। সবশেষে প্রথম যোগফলকে পরের বিয়োগফল দিয়ে ভাগ করে ফলাফল A-তে রাখা হবে।

কিন্তু একটি লঘু-বন্ধনীর মধ্যে অপর একটি লঘু-বন্ধনী থাকলে কি ভাবে এগোনো যাবে তা নীচের উদাহরণে দেখানো হচ্ছে।

উদাহরণ 8.

10 A = (B + (B*B – 4*A*C) ↑ 0.5) / (2*A)

এই উদাহরণে তিনটি লঘু-বন্ধনী ব্যবহার করা হয়েছে। প্রথমে সবচেয়ে ভিতরের লঘু-বন্ধনীর কাজ করতে হবে। সেই লঘু-বন্ধনীর ক্ষেত্রে প্রথমে B-এর মানের সঙ্গে আর একবার B-এর মান গুণ করে সেই গুণফল থেকে 4 সংখ্যার সঙ্গে A-এর মান এবং তার সঙ্গে আবার C-এর মান গুণ করে মোট গুণফল বিয়োগ করতে হবে। এরপর বিয়োগফলটির 0.5-এর সূচক বের করে বাইরের লঘু-বন্ধনীর কাজ শুরু করা যাবে। B-এর মানের সঙ্গে এই ফল এখন যোগ হবে। A চলরাশির নূতন মানটি নির্ণয়ের ক্ষেত্রে আরও কিছু করণীয় আছে। 2-এর সঙ্গে A-এর আগের মানের গুণফল নির্ণয় করা দরকার। এই গুণফল দিয়ে আগের যোগফলটিকে ভাগ করলে যে মানটি বেরোবে, সেটিই A চলরাশির নূতন মান হবে। মনে করা যাক, 10 সংখ্যক লাইনটির নির্দেশ অনুসারে কাজ শুরু করার আগে A, B এবং C তে যথাক্রমে 2,4 এবং 2 আছে। তাহলে কমপিউটার কি ভাবে ডান পাশের মান বের করবে তা উপরের নিয়মানুসারে বের করে দেখানো যেতে পারে। প্রথমে B-কে B দিয়ে গুণ করার অর্থ 4 দিয়ে 4- কে গুণ অর্থাৎ গুণফল হবে 16। এরপর 4-কে A-এর মান দিয়ে গুণ করলে পাওয়া যাবে 8 এবং এই গুণফলটিকে C-এর মান দিয়ে গুণ করার জন্যে ফল হবে 16। 16 সংখ্যাটিকে গুণফল 16 থেকে বাদ দিলে হবে 0। অর্থাৎ এরপর 0 ↑ 0.5 করলেও লব্ধফল পাওয়া যাবে শূন্যই। কাজেই B-এর মানের সঙ্গে 0 যোগ করলে যোগফল সেই 4-ই থাকবে। এবারে 2-কে A-এর মান দিয়ে গুণ করা দরকার। সেক্ষেত্রে গুণফল 4। তাহলে শেষ পর্যন্ত 4-কে 4 দিয়ে ভাগ করলে যে ভাগফল বেরোবে A-এর নূতন মান হবে তাই এবং এক্ষেত্রে 1। A-এর পুরনো মানটিকে এই নতুন মানটির জন্য জায়গা ছেড়ে দিতে হবে। অর্থাৎ এরপর A-এর মান ব্যবহার করলেই তা হবে 1।

উদাহরণ 9.

10 N% = N% + 1

এই নির্দেশ অনুসারে কাজ করে N%-এ যে পূর্ণ সংখ্যা ছিল তার সঙ্গে 1 যোগ করে N%-এর নূতন মান পাওয়া যাবে। 10 সংখ্যক লাইনের নির্দেশ করার আগে N%-এর মান 4 থাকলে এই নির্দেশ করার পর N%-এ 5 থাকবে।

উদাহরণ 10.

LET নির্দেশের সাহায্যে সংখ্যা ছাড়াও বর্নমালা সূচক ধ্রুবক কোনো চলরাশির মান হিসেবে লেখা যেতে পারে। যেমন,

15 LET A$ = 'BASIC PAY'

এক্ষেত্রে 8টি অক্ষর A$-এ রাখা হবে। BASIC-এর B থেকে আরম্ভ করে PAY-এর Y পর্যন্ত মোট আটটি অক্ষর আছে, C এবং P-এর মধ্যে যে ফাঁকা অক্ষরটি আছে সেটিও ওই আটটি অক্ষরের একটি। প্রোগ্রামে এর পর A$ ব্যবহার করলেই ওই আটটি অক্ষর পাওয়া যাবে।

READ - নির্দেশ ঃ

LET নির্দেশ যেমন একটি চলরাশির মান আরোপ করার জন্য ব্যবহার করা হয়ে থাকে, READ নির্দেশটিও তেমনি একটি চলরাশির মান আরোপ করার জন্য ব্যবহৃত হবে। অর্থাৎ উভয়ের কাজের ধরন এক, তবে READ-এর ক্ষেত্রে একটি নির্দেশের সাহায্যে একাধিক চলরাশির মান আরোপ করা যায়। এই নির্দেশটি লেখার ছক ঃ

লাইন সংখ্যা READ চলরাশির তালিকা

যেমন,

উদাহরণ 1.

10 READ A, B, C

চলরাশির তালিকায় একের বেশি চলরাশি থাকলে চলরাশিগুলিকে আলাদা করে দেখানোর জন্য কমা চিহ্ন ব্যবহার করা হয়। চলরাশির তালিকায় বিভিন্ন ধরনের চলরাশির নাম থাকা সম্ভব। যেমন, সংখ্যা-সূচক কিংবা বর্ণমালা-সূচক।

উদাহরণ 2.

15 READ A, D$

কোনো প্রোগ্রামে একটি READ নির্দেশ থাকলে এক বা একাধিক DATA নির্দেশও থাকবে। DATA নির্দেশগুলির তালিকার প্রথম ধ্রুবকটিকে READ নির্দেশের তালিকার প্রথম চলরাশির মান হিসেবে ধরা হবে। এরপর দ্বিতীয় ধ্রুবক হবে দ্বিতীয় চলরাশির মান। এইভাবে DATA নির্দেশের ধ্রুবকের তালিকার সঙ্গে READ নির্দেশের চলরাশির তালিকার সম্বন্ধ আরোপ করা হয়। একটি READ নির্দেশে যে সব চলরাশি থাকে তার মান এক বা একাধিক DATA নির্দেশে থাকা সম্ভব। আবার একাধিক READ নির্দেশের চলরাশির মান একটি DATA নির্দেশেও থাকতে পারে। দৃষ্টান্তের সাহায্যে ব্যাপারটা তুলে ধরা যাক।

উদাহরণ 3.

10 DATA 2.5

```
15 DATA 3.6, 1.8
20 READ A, B, C
```

এখানে 20 সংখ্যক লাইনে READ নির্দেশের তালিকায় তিনটি চলরাশির নাম দেওয়া আছে। এই তিনটি চলরাশির মান 10 সংখ্যক এবং 15 সংখ্যক লাইনের DATA নির্দেশে পাওয়া যাবে। READ নির্দেশে চলরাশির নামের তালিকায় A-এর নাম প্রথমে আছে এবং DATA নির্দেশগুলির প্রথম ধ্রুবকটি 2.5 হওয়াতে, A-এর মান হবে 2.5। এরপর B এবং C-এর মান হবে যথাক্রমে 3.6 এবং 1.8। এখানে একটি READ নির্দেশের তিনটি চলরাশির মান দুটি DATA নির্দেশে আছে।

উদাহরণ 4.

```
10 DATA 2.5, 3.6, 1.8
20 READ A, B
30 READ C
```

এবারে দুটি READ নির্দেশের চলরাশির মান একটি DATA নির্দেশের অন্তর্ভুক্ত।

উদাহরণ 5.

```
10 DATA 2.5, 3.6
20 READ A, B, C
```

এই দৃষ্টান্তটি ভিন্ন ধরনের। এখানে READ নির্দেশে চলরাশির তালিকায় চলরাশির সংখ্যা তিন, কিন্তু DATA নির্দেশে ধ্রুবকের তালিকায় কেবলমাত্র দুটি ধ্রুবক আছে। কাজেই A এবং B-এর মান আরোপ করার পর C-এর মান আরোপ করতে গেলে দেখা যাবে C-এর মানের জন্য কোনো ধ্রুবক নির্দিষ্ট নেই। এক্ষেত্রে কমপিউটার 'OUT OF DATA' শব্দ তিনটি ছাপিয়ে থেমে যাবে।

উদাহরণ 6.

```
10 DATA 2.5, 3.6, 1.8
20 READ A, B
```

এই দৃষ্টান্তটি উদাহরণ 5-এর অনেকটা বিপরীত। এক্ষেত্রে READ-এর চলরাশির তালিকায় দুটি চলরাশির নাম আছে, কিন্তু DATA নির্দেশে ধ্রুবকের তালিকায় ধ্রুবকের সংখ্যা তিন। এখানে A এবং B-এর মান আরোপ করার পরে কমপিউটার পরের নির্দেশ পালন করবে। অর্থাৎ READ নির্দেশের চলরাশির নামের সংখ্যা DATA নির্দেশের ধ্রুবকের সংখ্যার চেয়ে কম থাকলে কমপিউটার অতিরিক্ত ধ্রুবকগুলি অগ্রাহ্য করে পরের নির্দেশগুলি ঠিকমত পালন করবে। সুতরাং উদাহরণ 6-এ 1.8 ধ্রুবকটি কাজে লাগছে না। তবে এমন

হওয়া সম্ভব যে কয়েকটি নির্দেশ পালন করার পর আবার একটি READ নির্দেশ থাকতে পারে। সেক্ষেত্রে READ নির্দেশে যে চলরাশির নাম থাকবে তার মান হবে 1.8।

উদাহরণ 7.

```
10 DATA 2500, AMAL
20 READ BPAY, NAME$
```

এখানে READ নির্দেশে চলরাশির তালিকায় দু ধরনের চলরাশির নাম দেওয়া আছে। তবে READ-এ যে ধরনের চলরাশির নাম থাকবে DATA নির্দেশেও সেই একই ধরনের ধ্রুবক দিতে হবে। তা না হলেই কমপিউটার 'TYPE MISMATCH' বলে থেমে যাবে। এই উদাহরণে প্রথম চলরাশিটি সংখ্যা-সূচক, কাজেই DATA নির্দেশের প্রথম ধ্রুবকটিও সংখ্যা-সূচক। এরপর READ-এ দ্বিতীয় চল-রাশির নামের শেষে $ চিহ্ন থাকায় এটি একটি সারি চলরাশি, কাজেই DATA নির্দেশের দ্বিতীয় ধ্রুবকটি একটি বর্ণমালা-সূচক ধ্রুবক।

DATA-নির্দেশ ঃ

একটি প্রোগ্রামে READ নির্দেশের সাহায্যে যে-সব চলরাশির মান পড়া হবে সে-সব মান সরবরাহ করা হয় DATA নির্দেশের সাহায্যে। একটি প্রোগ্রামে এক বা একাধিক DATA নির্দেশ থাকতে পারে। এই নির্দেশগুলি থাকে সাধারণত প্রোগ্রামের প্রথম দিকে কিংবা একেবারে শেষের দিকে। তবে যে-সব কমপিউটারে BASIC ইন্টারপ্রিটার ব্যবহার করা হয় সেখানে সাধারণত DATA নির্দেশ প্রথমেই দেওয়া হয়ে থাকে। এই নির্দেশটি লেখার ছক ঃ

লাইন সংখ্যা DATA ধ্রুবকের তালিকা

যেমন,

উদাহরণ 1.

```
10 DATA 10.5, 2.6
```

DATA নির্দেশগুলিকে ইন্টারপ্রিটার বা কমপাইলার ওই যন্ত্রের ভাষায় কোনো কাজ করার জন্য কোনো নির্দেশের রূপ দেয় না, কেবলমাত্র এই নির্দেশগুলির তালিকার ধ্রুবকগুলিকে স্মৃতিতে পরপর সাজিয়ে রাখার ব্যবস্থা করে। অর্থাৎ READ নির্দেশের জন্য যেমন ওই কমপিউটারের যন্ত্রের ভাষায় একটি নির্দেশ পাওয়া যাবে এবং সেই নির্দেশ পালন করে একটি চলরাশির মান পাওয়া সম্ভব, DATA নির্দেশের জন্য সেরকম কোনো যন্ত্রের ভাষায় নির্দেশ পাওয়া যাবে না।

DATA নির্দেশেও READ নির্দেশের মত একের বেশি ধ্রুবক

থাকলে ধ্রুবকগুলিকে আলাদা করে দেখানোর জন্য কমা চিহ্ন ব্যবহার করা হয়। এই নির্দেশে যে কোনো ধরনের ধ্রুবকই থাকতে পারে কিন্তু কোনোপ্রকার রাশি বা রাশিমালা থাকতে পারে না।

উদাহরণ 2.

10 DATA 3 + 4, 5.2 + 8.3

এখানে DATA নির্দেশে ধ্রুবকের বদলে দুটি ধ্রুবককে যোগ করা হয়েছে। এভাবে লিখলে ভুল হবে এবং ইন্টারপ্রিটার বা কমপাইলার এই ভুলের জন্য থেমে যাবে।

যদি বর্ণ ও সংখ্যামালা-সূচক ধ্রুবকে কোনো কমা চিহ্ন, কোলন চিহ্ন অথবা প্রথমে এবং শেষে কোনো ফাঁকা জায়গা রাখার প্রয়োজন না হয়, তাহলে ওই ধ্রুবক DATA নির্দেশে দুপাশের উক্তি চিহ্ন ছাড়া দেওয়া যায়।

উদাহরণ 3.

10 DATA BASIC PAY, NAME

20 READ A$, B$

এখানে A$ নামের স্মৃতিকোষে BASIC PAY এবং এই শব্দ দুটির মধ্যেকার ফাঁকা জায়গাটি ধরে মোট 9টি অক্ষর রাখা হবে এবং B$ নামের স্মৃতিকোষে NAME এই 4টি অক্ষর থাকবে। এই DATA নির্দেশের তালিকায় বর্ণ ও সংখ্যামালা-সূচক ধ্রুবক থাকা সত্ত্বেও কোনো উদ্ধৃতি চিহ্নর প্রয়োজন নেই। কারণ প্রথম ধ্রুবকটিতে ফাঁকা জায়গাটি BASIC এবং PAY-র মধ্যে রয়েছে এবং দ্বিতীয় ধ্রুবকেও কোনো ফাঁকা জায়গা বা কোনো কমা কিংবা কোলন চিহ্ন নেই।

কোনো প্রোগ্রামে READ নির্দেশ থাকলে অতি অবশ্যই DATA নির্দেশও থাকবে। একটি প্রোগ্রামে একাধিক DATA নির্দেশ থাকতে পারে। DATA নির্দেশের সবকটি ধ্রুবক প্রোগ্রামে কাজে না লাগালে কোনো ভুল হবে না, একথা আগে বলেছি। DATA নির্দেশে ধ্রুবকগুলি একের পর এক কমার সাহায্যে আলাদা করে রাখা থাকে। এই ধ্রুবকগুলির মধ্যিখানের কোনো একটি ধ্রুবককে পড়তে হলে এর আগের সবকটি ধ্রুবক পড়ার পরই তা সম্ভব।

PRINT-নির্দেশ ঃ

ভিডিইউর পর্দায় কিছু ফুটিয়ে তোলার জন্য এই নির্দেশটি ব্যবহার করা হয়। PRINT নির্দেশ নানাভাবে লেখা সম্ভব। যেমন,

1. লাইন সংখ্যা PRINT

এখানে PRINT-এর পরে আর কিছু না থাকায় এই নির্দেশ

পালন করার অর্থ, পর্দায় একটি লাইনে কিছু না লিখেই কমপিউটার পরের লাইনে চলে যাবে। অর্থাৎ এই নির্দেশের সাহায্যে একটি ফাঁকা লাইন লেখা হবে।

2. লাইনসংখ্যা PRINT রাশিমালার তালিকা

কোনো রাশির মান দেখার প্রয়োজন হলে এই নির্দেশটি দেওয়া হয়।

উদাহরণ 1.

50 PRINT A, B

এই নির্দেশটির অর্থ A রাশির মান লেখার পরে B-এর মান লেখা হবে। কিন্তু এদের মান লাইনের কোথায় লেখা হবে ? এখানে A এবং B নাম দুটি দেখেই বোঝা যায়, এরা একক-দৈর্ঘ্য চলরাশি। সুতরাং এদের মান 7 অঙ্কের চেয়ে কখনোই বড় হবে না। A ধনাত্মক হলে লাইনের প্রথম স্থানটি ফাঁকা থাকবে, এরপর A-এর মানের অঙ্কগুলি লেখা হবে। কিন্তু B ঋণাত্মক হলে 15 সংখ্যক স্থানে বিয়োগ-চিহ্ন লিখে B-এর মানটি লিখতে হবে। সংখ্যার ক্ষেত্রে ধনাত্মক সংখ্যার বেলায় অঙ্কগুলি লেখার আগে এবং পরে একটি করে ফাঁকা স্থান থাকবে। কিন্তু ঋণাত্মক হলে শুরুতে কোনো ফাঁকা স্থান হয় না, কিন্তু শেষে একটি ফাঁকা স্থান থাকবে।

বেসিক পদ্ধতি একটি লাইনকে কয়েকটি অঞ্চলে ভাগ করে এবং প্রত্যেকটি অঞ্চলে 14টি অক্ষর লেখার স্থান থাকে। সাধারণত একটি লাইনে 5টি অঞ্চল থাকে। PRINT নির্দেশের পরে একের বেশি রাশিমালা থাকলে একটি থেকে আর একটিকে পৃথক করার জন্য কতগুলি চিহ্ন ব্যবহার করে রাশির মানগুলি লেখার সময় একটার সঙ্গে আর একটার মধ্যে কতটা জায়গা ফাঁকা থাকবে তা বোঝানো হয়। যেমন,

পৃথক চিহ্ন	লাইনে কিভাবে লেখা হবে
কমা	পরের মান পরের অঞ্চলের শুরুতে
সেমিকোলন	সংখ্যার ক্ষেত্রে আগেরটি শেষ হওয়ার পর একটি স্থান ফাঁকা রেখে পরের মান লেখা শুরু হবে। ধনাত্মক হলে আরও একটি ফাঁকা জায়গা থাকবে। '+' চিহ্ন যেহেতু লেখা হয় না। সারি চলরাশির ক্ষেত্রে দুটির মধ্যে কোনো ফাঁকা জায়গা থাকে না।

উদাহরণ 2.

50 PRINT A, B

এখন A এবং B-এর মান যথাক্রমে 3.1 এবং –3456.72 হলে এই

নির্দেশটির জন্য পর্দায় একটি লাইনের কোন জায়গাতে কি লেখা হবে তা নীচে দেখানো হচ্ছে।

```
              1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2
1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8
  3 . 1                     - 3 4 5 6 . 7 2
```

PRINT নির্দেশে A এবং B- কে কমা চিহ্নের সাহায্যে পৃথক করা হয়েছে। ফলে A এবং B-এর মান দুটি অঞ্চলে লেখা হবে। অর্থাৎ লাইনের 2-সংখ্যক স্থান থেকে A-এর মান লেখা হবে। উপরের ছবিতে তাই 2-এর নীচে 3 লেখা হয়েছে। এরপর 3-সংখ্যক স্থানের নীচে দশমিক চিহ্নটি এবং 4-সংখ্যক স্থানের নীচে 1 লেখা হবে। B-এর মান দ্বিতীয় অঞ্চলে লেখা হবে। কাজেই লাইনে 5-সংখ্যক স্থান থেকে 14-সংখ্যক স্থান ফাঁকা থাকবে। এরপর B ঋণাত্মক হওয়ায় এর মান 15-সংখ্যক স্থান থেকে লেখা শুরু হবে। অর্থাৎ বিয়োগ চিহ্নটি (–) 15-সংখ্যক স্থানে লেখা হবে এবং এরপরের স্থান থেকে অঙ্কগুলি লেখা হবে। B ধনাত্মক হলে কিন্তু 15-সংখ্যক স্থানটি ফাঁকা থাকতো।

উদাহরণ 3.

```
50 PRINT A; B
```

এখানে A এবং B-কে পৃথক করার জন্য সেমিকোলন ব্যবহার করা হয়েছে। কাজেই এবারে A-এর মান লেখা যে স্থানে শেষ হয়েছে তারপরে একটি জায়গা ফাঁকা রেখে ঠিক তার পরের স্থান থেকেই B-এর মান লেখা শুরু হবে। উদাহরণ 2-এর মত A এবং B-এর মান একই হলে এবারে লাইনের কোন জায়গাতে কি লেখা হবে তা নীচে দেখানো হচ্ছে।

```
                  1 1 1 1 1 1 1
1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6
  3 . 1   - 3 4 5 6 . 7 2
```

এবারে কিন্তু A এবং B-এর মান দুটি ভিন্ন অঞ্চলে লেখা হয় নি। A-এর মান লেখা শেষ হওয়ার পর একটি জায়গা ফাঁকা রেখেই B-এর মান লেখা শুরু হয়েছে। B অবশ্য ধনাত্মক হলে 6-সংখ্যক স্থানটিও ফাঁকা থাকতো। এখানে B ঋণাত্মক বলে বিয়োগ চিহ্নটি 6-সংখ্যক স্থানে লেখা হয়েছে।

PRINT নির্দেশে কেবলমাত্র চলরাশির নাম ব্যবহার ছাড়াও রাশি বা রাশিমালা থাকতে পারে। যেমন,

উদাহরণ 4.

```
60 PRINT 5.2 + 8.3
```

এক্ষেত্রে পর্দায় লেখার আগে সংখ্যা দুটি যোগ করে যোগফলটি লেখা হবে।

উদাহরণ 5.

```
60 PRINT A + B
```

এবারে A এবং B এই দুটি চলরাশির মান যোগ করে যোগফলটি লেখা হবে।

অনেক সময়ে পর্দায় একাধিক চলরাশির মান লেখা থাকলে কোন চলরাশির মান কোনটি তা বোঝার জন্য আবার প্রোগ্রামটির PRINT নির্দেশ দেখতে হবে। সেইজন্য লেখার সময়েই কোনটা কার মান তা বোঝানোর ব্যবস্থা করাই সুবিধাজনক।

উদাহরণ 6

```
10 PRINT "THE VALUE OF A= "; A, " THE VALUE OF B= "; B"
```

এই PRINT নির্দেশটি কি করবে ? প্রথমে THE VALUE OF A = লিখবে। এর জন্য কটি জায়গার প্রয়োজন ? প্রতিটি শব্দের পর একটি করে ফাঁকা জায়গা ধরে এবং সমান চিহ্নের পর উদ্ধৃতি চিহ্নের আগে একটি ফাঁকা জায়গা আছে ধরে নিলে মোট জায়গার প্রয়োজন 16টি। এই 16টি জায়গার পরে A-এর মান লেখা হবে। সেমিকোলন থাকার জন্য A ঋণাত্মক হলে ঠিক 17-সংখ্যক স্থান থেকেই A-এর মান লেখা শুরু হবে। ধনাত্মক হলে A-এর মান একটি জায়গা ফাঁকা রেখে 18-সংখ্যক স্থান থেকে লিখতে হবে। A একটি একক-দৈর্ঘ্য সংখ্যা রাখা যায় এমন একটি চলরাশি হওয়ায় এর মান যত বড়ই হোক না কেন তা 7টি অঙ্কের বেশি কখনোই হবে না। কাজেই A-এর মান লেখা 28-সংখ্যক স্থানের কিছু আগেই শেষ হবে। PRINT নির্দেশে A-এর পর কমা থাকায় 'THE VALUE OF B =' লেখা শুরু হবে ঠিক 29-সংখ্যক স্থান থেকে। অর্থাৎ 'THE VALUE OF B =' -এর T অক্ষরটি 29-সংখ্যক স্থানে লেখা হবে অর্থাৎ কমা চিহ্ন থাকায় এবারে একটি অঞ্চল থেকে লেখা শুরু করতে হবে। 28-সংখ্যক স্থানের আগেই A-এর মান শেষ হওয়ায় এটি তৃতীয় অঞ্চল থেকেই শুরু হবে এবং তারপর অন্যান্য অক্ষরগুলি ঠিক যেমন ভাবে উদ্ধৃতির চিহ্নর মধ্যে লেখা রয়েছে তেমনিভাবে পর পর লেখা চলবে। এরপর B-এর মান লেখা একটি স্থান ফাঁকা রেখে শুরু হবে কিনা তা নির্ভর করবে B ধনাত্মক না ঋণাত্মক তার উপরে।

উদাহরণ 7.

```
10 A# = 6/7
20 B = 6#/7
30 C# = 6#/7
```

```
40 D = 6/7
50 PRINT A# ; B, C#, D
60 END
```

PRINT নির্দেশে প্রথম চলরাশির নামের শেষে সংখ্যা-চিহ্ন (#) দেখেই বোঝা যাচ্ছে এটি একটি দ্বি-দৈর্ঘ্য সংখ্যা। এর মান 10-সংখ্যক লাইনে বের করা হয়েছে এবং তা হল .8571428656578 064। এর জন্য মোট 19টি জায়গার প্রয়োজন। 19টি কেন ? প্রথম কথা, সংখ্যাটি ধনাত্মক হওয়াতে প্রথম জায়গা অর্থাৎ 1-সংখ্যক স্থানটি ফাঁকা থাকবে। এরপর দশমিক চিহ্ন সহ সংখ্যাটির জন্য 17টি জায়গার প্রয়োজন। সবশেষে একটি সংখ্যা লেখার পরে একটি জায়গা ফাঁকা থাকে। কাজেই সব মিলিয়ে হল 19টি। এরপর B-এর মান ছাপাতে বলা হয়েছে। A# এবং B-এর মধ্যে সেমিকোলন চিহ্ন ব্যবহার করায় B-এর মান শুরু হবে 19টি স্থানের ঠিক পরেই। B-এর মান ধনাত্মক হওয়ায় 20-সংখ্যক স্থানটি ফাঁকা থাকবে এবং 21-সংখ্যক স্থান থেকে দশমিক চিহ্ন এবং 7টি অঙ্ক ছাপাতে মোট 8টি স্থানের প্রয়োজন হবে। এখানে বলে রাখা ভাল যে, B একটি একক-দৈর্ঘ্য সংখ্যা। কাজেই B-এর জন্য 7টি অঙ্ক হবে। B-এর মান লেখার পরে একটি ফাঁকা জায়গা থাকবে। কাজেই B-এর মান লেখা শেষ করে ফাঁকা জায়গা নিয়ে 29-সংখ্যক স্থান পর্যন্ত চলে যাবে। B এবং C# এর মধ্যে কমা চিহ্ন থাকায় C#-এর মান ছাপানো শুরু হবে লাইনে একটি নতুন অঞ্চল থেকে। একটি অঞ্চলে যে 14টি স্থান থাকে তা আগেই বলা হয়েছে। B-এর মান লেখা শেষ করতে তৃতীয় অঞ্চলের একটি স্থান নিয়ে নেওয়াতে C#-এর মান লেখা শুরু হবে চতুর্থ অঞ্চল থেকে অর্থাৎ 43-সংখ্যক স্থান থেকে। C# রাশিটি আবার দ্বি-দৈর্ঘ্য চলরাশি হওয়ায় 19টি জায়গার প্রয়োজন। সেক্ষেত্রে চতুর্থ অঞ্চলটি পুরো এবং পঞ্চম অঞ্চলের কিছু স্থান এই চলরাশিটির মান লেখার জন্য দরকার। এরপর C# এবং D-এর মধ্যে কমা চিহ্ন থাকায় D-এর মান আর ওই একই লাইনে না লিখে পরের লাইনের প্রথম অঞ্চলে লেখা হবে। এর কারণ একটি লাইনে কেবলমাত্র 5টি অঞ্চলই লেখা সম্ভব।

LPRINT—নির্দেশঃ

PRINT নির্দেশ দিলে ভিডিইউর পর্দায় অক্ষর বা সংখ্যা ফুটে ওঠে। কিন্তু ডট ম্যাট্রিক্স প্রিন্টারে যে কাগজ লাগানো থাকে সেখানে ছাপানোর প্রয়োজন হলে PRINT নিদের্শের বদলে LPRINT নির্দেশ দিতে হবে।

LPRINT নির্দেশের সাহায্যে প্রিন্টারের সঙ্গে জড়ানো কাগজের

প্রতিটি লাইনে 80টি অক্ষর লেখা যায়। এর চেয়ে বেশি অক্ষরও এক লাইনে লেখা সম্ভব। তবে সেক্ষেত্রে অপর একটি নির্দেশের সাহায্য নিতে হয়। এ ব্যপারে এবং PRINT ও LPRINT সম্বন্ধে আরও বিশদ আলোচনা পরে করা হবে।

PRINT নির্দেশের সাহায্যে যে ভাবে ভিডিইউ-এর পর্দায় লেখা ফুটে ওঠে এবং এই নির্দেশে যা যা করা সম্ভব বলা হয়েছে সে সবই LPRINT নির্দেশও করা যায়।

END—নির্দেশেঃ

একটি প্রোগ্রামের সব কটি নির্দেশ করার পরে থামার জন্য এই নির্দেশটি দেওয়া হয়। সাধারণত একটি প্রোগ্রামের সব শেষে এই নির্দেশটি দেওয়া থাকে। এটি পাওয়ার পর OK শব্দটি পর্দায় ফুটে উঠবে এবং কমপিউটারটি থেমে থাকবে। এরপর এই প্রোগ্রামটি চালিয়ে ফলাফল বের করা যাবে। PRINT নির্দেশের উদাহরণ 7-এ এই END নির্দেশের ব্যবহার দেখানো হয়েছে।

INPUT - নির্দেশঃ

READ নির্দেশের কথা আগেই আলোচনা করা হয়েছে। এই READ-নির্দেশের সাহায্যে যে-সব রাশির মান পড়া হয় সে-সব রাশির মান ওই প্রোগ্রামের মধ্যেই DATA নির্দেশের সাহায্যে সরবরাহ করা হয়। এতে অসুবিধে এই যে, যদি ওইসব রাশির মানের পরিবর্তনের প্রয়োজন হয় তাহলে DATA নির্দেশের পরিবর্তনের দরকার হবে। কিন্তু আমরা একথা আগেই আলোচনা করেছি যে, কোনো একটি সমস্যা সমাধানের জন্য এমনভাবে নির্দেশগুলি দেওয়া দরকার যাতে যে কোনো তথ্যের জন্যই ওই নির্দেশগুলির পরিবর্তনের প্রয়োজন না ঘটে। READ নির্দেশ দিলেই DATA নির্দেশও থাকবে। সেক্ষেত্রে যতবার নতুন তথ্যের প্রয়োজন হবে ততবারই DATA নির্দেশ বদলানোর দরকার। কিন্তু এই দুটি নির্দেশের বদলে যদি INPUT নির্দেশটি লেখা হয় তাহলে তার প্রয়োজন হবে না। উদাহরণের সাহায্যে ব্যপারটা বোঝানো যাক।

উদাহরণ 1.

```
10 INPUT A, B
20 C = A + B
30 PRINT C
40 END
```

READ নির্দেশের মতই INPUT নির্দেশটি A এবং B-এর মান

পড়বে। READ-এর বেলায় A এবং B-এর মান প্রোগ্রামের মধ্যেই DATA নির্দেশে দেওয়া থাকবে। কিন্তু INPUT-এর ক্ষেত্রে এই মান কি করে পাওয়া যাবে? কমপিউটার এই নির্দেশটি পালন করার সময় ভিডিইউ-এর পর্দায় জিজ্ঞাসার চিহ্ন (?) লিখে থেমে থাকবে। এরপর A-এর মান কী-বোর্ডের মাধ্যমে লিখে কমা দিয়ে আবার B-এর মান লিখতে হবে। এরপর RETURN কী টিপলে তবেই ওই সংখ্যা দুটি A এবং B-এর মান হিসেবে ধরে নিয়ে পরের লাইনে গিয়ে সেখানকার নির্দেশ পালন করবে। এরপর আবার যদি A এবং B-এর নতুন কোনো মানের জন্য প্রোগ্রামটি করার দরকার হয় তবে কেবলমাত্র ওই একই প্রোগ্রাম চালিয়ে তা করা সম্ভব হবে। প্রোগ্রামটি আবার করে চালালেই INPUT নির্দেশটি করতে গিয়ে কমপিউটার জিজ্ঞাসার চিহ্ন (?) লিখে থেমে যাবে। এরপর A এবং B-এর মান কমা চিহ্নর সাহায্যে আলাদা করে দিয়ে RETURN কী টিপলেই কমপিউটার প্রথম বারের মতই C-এর মান লিখে থেমে যাবে। কাজেই যতবারই A এবং B-এর নতুন মানের জন্য প্রোগ্রামটি চালানোর প্রয়োজন হোক না প্রোগ্রামের নির্দেশের কোন পরিবর্তনের দরকার হবে না।

READ নির্দেশের মতই INPUT নির্দেশেও একাধিক চলরাশির নাম থাকতে পারে এবং সেক্ষেত্রে একটির সঙ্গে অন্যটির পার্থক্য আনার জন্য কমা-চিহ্ন ব্যবহার করা হয়।

একটি INPUT নির্দেশে আবার READ-এর মতই বিভিন্ন ধরনের চলরাশির নাম থাকতে পারে। দৃষ্টান্তস্বরূপ উদাহরণ 2 লক্ষ্য করা যাক।

উদাহরণ 2.

```
10 INPUT A, B$, C, D$
```

এক্ষেত্রে A ও C দুটি সংখ্যা চলরাশি এবং B$ ও D$ দুটি সারি চলরাশি। কাজেই এদের মান দেওয়ার সময় মনে রাখতে হবে যে প্রথম এবং তৃতীয়টি সংখ্যা, দ্বিতীয় ও চতুর্থটি বেসিক বর্ণমালার যে কোনো অক্ষর হতে পারে। DATA নির্দেশে সারি চলরাশির মান দেওয়ার যেসব নিয়ম রয়েছে তা এক্ষেত্রেও প্রযোজ্য।

অনেক ক্ষেত্রে INPUT A, B দেওয়া থাকলে প্রোগ্রামটি চলাকালীন মান চাওয়া হলে A কিংবা B কোন চলরাশিটি INPUT নির্দেশে আগে আছে মনে না রইলে মান দিতে ভুল হওয়ার একটা আশঙ্কা থেকে যায়। অবশ্য নীচের পদ্ধতি অনুসরণ করলে যে কোনো রকম ভুলের আশঙ্কা এড়িয়ে যাওয়া সম্ভব।

উদাহরণ 3.

```
10 INPUT "A ="; A
```

```
20 INPUT "B="; B
30 C = A + B
40 PRINT C
50 END
```

এবারে এই প্রোগ্রামটি চালানোর সময়ে INPUT নির্দেশটি পালন করার অর্থ হবে ভিডিইউ-এর পর্দায় একটি লাইনে A = এই দুটি অক্ষর বা চিহ্নের পর একটি জিজ্ঞাসা চিহ্ন (?) লিখে থেমে থাকবে। কাজেই ভুল হওয়ার কোনো আশঙ্কা থাকছে না। এবারে A-এর মান কী-বোর্ডের মাধ্যমে দিয়ে RETURN কী টিপলেই পর্দায় পরের লাইনে 20 সংখ্যক লাইনের নির্দেশ অনুযায়ী B = ? লিখে থেমে যাবে। এরপর আবার B-এর মান আগের মতই কী-বোর্ডের মাধ্যমে দিয়ে RETURN কী টিপলে কমপিউটার পরের নির্দেশগুলি পালন করে পর্দায় পরের লাইনে C-এর মান ছাপিয়ে এবারে একেবারেই থেমে যাবে।

INPUT নির্দেশে কমপিউটার দুটি উদ্ধৃতি-চিহ্নর মধ্যেকার সমস্ত অক্ষর বা চিহ্ন পর্দায় হুবহু লিখে একটি জিজ্ঞাসা চিহ্ন দিয়ে থেমে যায়। তবে একথা মনে রাখা দরকার যে, একটি INPUT নির্দেশে একটির বেশি উদ্ধৃতি চিহ্ন ব্যবহার করা যায় না। অর্থাৎ উদাহরণ 3-এ 10 সংখ্যক এবং 20 সংখ্যক লাইন দুটি একত্র করে একটি INPUT নির্দেশ লিখলে ভুল হবে।

10 সংখ্যক এবং 20 সংখ্যক লাইনে INPUT নির্দেশে সেমিকোলন চিহ্নের বদলে কমা চিহ্ন ব্যবহার করলে কি পরিবর্তন হবে? সেমিকোলন চিহ্ন দেওয়াতে উদ্ধৃতি-চিহ্নের মধ্যেকার অক্ষরগুলি লেখার ঠিক পরই একটি জিজ্ঞাসা-চিহ্ন থাকবে। কিন্তু কমা চিহ্নর বেলাতে ওই জিজ্ঞাসা-চিহ্নটি থাকবে না। তবে কমা চিহ্নের জন্য সংখ্যাটি পরের অঞ্চল থেকেও হবে না (যেমন PRINT নির্দেশের ক্ষেত্রে হয়ে থাকে), উদ্ধৃতি-চিহ্নর মধ্যের শেষ অক্ষরটি যেখানে শেষ হবে ঠিক তার পর থেকেই সংখ্যাটি লেখা যাবে।

REM-নির্দেশঃ

এই নির্দেশটি ইংরেজি REMARK শব্দর সংক্ষিপ্ত রূপ। প্রোগ্রামে মন্তব্য দেওয়ার জন্য এর প্রয়োজন। প্রোগ্রামটিতে কি ধরনের সমস্যার সমাধান করা হয়েছে তা এই নির্দেশের মাধ্যমে লিখে রাখা যায়। প্রোগ্রামটি লেখার অনেকদিন পরে কেবলমাত্র এই নির্দেশটি দেখেই বোঝা যাবে, এটি কোন সমস্যা সমাধানের জন্য ব্যবহার করা সম্ভব। আসলে বেশ কিছুদিন আগের লেখা প্রোগ্রামের নির্দেশ দেখে বোঝা মুশকিল এটি কি ধরনের সমস্যার সমাধান

করতে সক্ষম। সেইজন্যই এই নির্দেশ। এই নির্দেশটি লেখার নিয়মঃ

লাইন সংখ্যা REM বেসিকের যে কোনো অক্ষর

উদাহরণ 1.

দুটি সংখ্যা পড়ে, যোগ করে, যোগফল ছাপানোর প্রোগ্রামে একটি REM নির্দেশ দিয়ে প্রোগ্রামটি যে সমস্যা সমাধানের জন্য করা হয়েছে তা বোঝানো যেতে পারে।

```
5 REM THIS PROGRAM FINDS THE SUM OF TWO NUMBERS
```

REM এই মূল শব্দটির পরে ওই লাইনে যা লেখা থাকবে সবটাই মন্তব্য হিসেবে ধরে নিতে হবে। এটিও 'DATA নির্দেশের মত। কমপিউটার প্রোগ্রামের নির্দেশ অনুসারে কাজ শুরু করে এই নির্দেশে এলে এখানে কিছু কাজ করার থাকে না। পুরো প্রোগ্রামটি ছাপালে অন্যান্য নির্দেশের সঙ্গে এটিও ছাপানো হয়। পরে এই ছাপানো লাইনটি দেখে বোঝা যাবে এই প্রোগ্রামটি কোন সমস্যার সমাধান করতে সক্ষম।

একটি প্রোগ্রামে একাধিক REM নির্দেশ দেওয়া সম্ভব এবং প্রোগ্রামে যে কোনো জায়গাতে তা লেখা যায়। এক লাইনে পুরো মন্তব্য লেখা না গেলে পরের লাইনে আবার লাইন সংখ্যা দিয়ে REM মূল শব্দটি লিখে তবেই বাকি মন্তব্য লেখা যাবে।

উদাহরণ 2.

```
10 REM THIS IS MY FIRST PROGRAM IN BASIC AND
15 REM THIS FINDS THE SUM OF TWO NUMBERS
```

অনেক সময়ে একটি বড় প্রোগ্রাম কতগুলি বিভিন্ন ধরনের কাজের সমষ্টি। প্রত্যেকটি ধরনের কাজের নির্দেশের আগে একটি করে REM-এর সাহায্যে মন্তব্য লিখে রাখা সম্ভব। এর ফলে এর পরের নির্দেশগুলিতে কি ধরনের কাজ করা হচ্ছে ভবিষ্যতে ওই মন্তব্য দেখেই তা বোঝা যাবে।

REM-এর বদলে কেবলমাত্র একটি উদ্ধৃতি-চিহ্নে দিয়ে ওই একই কাজ করা সম্ভব। এক্ষেত্রে উদ্ধৃতি-চিহ্ন দিয়ে শুরু, কিন্তু শেষে কোনো উদ্ধৃতি-চিহ্ন দেওয়া হয় না।

উদারহর। 3.

```
10 'THIS IS MY FIRST PROGRAM IN BASIC
```

এই উদ্ধৃতি-চিহ্নের সাহায্যে একটি লাইনের যে কোনো জায়াগাতেই মন্তব্য লেখা সম্ভব। উদ্ধৃতি-চিহ্ন পাওয়ার পর সেই লাইনের শেষ পর্যন্ত সব কিছুই মন্তব্য হিসেবে ধরা হয়।

উদাহরণ 4.

```
10 NETPAY
= BASIC + ALLOWACES–DEDUCTIONS' CALCULATE PAY
```

এখানে DEDUCTIONS-এর পরে উদ্ধৃতি-চিহ্নের সাহায্যে মন্তব্য করা হল যে এই লাইনে PAY হিসেব করা হবে। উদ্ধৃতি চিহ্ন দিলে সেই লাইনে আর মন্তব্য ছাড়া অন্য কোনো নির্দেশ দেওয়া যাবে না।

DATA নির্দেশে কিন্তু REM ব্যবহার করা যায় না। DATA নির্দেশে REM ব্যবহার করলে কমপিউটার এটি একটি সারি রাশির মান হিসেবে ধরে নেবে।

GO TO-নির্দেশঃ

কমপিউটার সাধারণত একটি প্রোগ্রামের প্রথম নির্দেশটি পালন করার পর দ্বিতীয় নির্দেশটি পালন করে। তারপর তৃতীয় লাইনের নির্দেশ অনুসরণ করে। এইভাবে পরপর নির্দেশ অনুসারে কাজ চলতে থাকে। পরপর নির্দেশগুলি পালন করার জন্য প্রোগ্রামে আলাদা ভাবে আর কোনো নির্দেশ দেওয়ার প্রয়োজন হয় না। কিন্তু অনেক সময়ে একটি সমস্যা সমাধান করতে গিয়ে দেখা যায় যে, পরপর নির্দেশগুলি পালন করার পরিবর্তে কয়েকটি লাইনের নির্দেশ বাদ দিয়ে তার পরের কিছু সংখ্যক লাইনের নির্দেশ পালন করে আবার যে-সব নির্দেশ বাদ দেওয়া হয়েছে সেগুলি করার প্রয়োজন। এ জন্য GO TO নির্দেশ ব্যবহার করা হয়। এই নির্দেশ লেখার ধরনঃ

লাইন সংখ্যা GO TO লাইন সংখ্যা

GO TO নির্দেশের ডান পাশে যে লাইন সংখ্যা থাকে তা দিয়ে বোঝানো হয়, GO TO নির্দেশটি পালন করার অর্থ সরাসরি ওই লাইন সংখ্যায় যাওয়া এবং সেই নির্দেশ থেকে শুরু করে পরপর নির্দেশগুলি পালন করা। GO এবং TO শব্দ দুটি একই সঙ্গে অথবা আলাদা করে দুভাবেই লেখা যায়।

উদাহরণ 1.

কয়েকটি জোড় সংখ্যার যোগফল বের করার জন্য নীচের প্রোগ্রামটির সাহায্য নেওয়া যেতে পারে।

```
10 INPUT A, B
20 C = A + B
30 PRINT C
40 GO TO 10
50 END
```

এই উদাহরণে প্রথমে INPUT নির্দেশের জন্য জিজ্ঞাসার চিহ্ন (?) দিয়ে কমপিউটার থেমে থাকবে। এবারে A এবং B-এর জন্য দুটি সংখ্যা দিয়ে RETURN কী টিপলে 20-সংখ্যক এবং 30-সংখ্যক লাইনের নির্দেশ পালন করে 40-সংখ্যক লাইনে এসে এই নির্দেশ অনুসারে আবার 10-সংখ্যক লাইনে গিয়ে সেখানকার নির্দেশ পালন করবে। এর অর্থ, আবার আগের মতই দুটি সংখ্যা পড়ে, যোগ করে, যোগফল ছাপিয়ে 10-সংখ্যক লাইনে চলে আসবে। এইভাবে চলতেই থাকবে। কিন্তু প্রোগ্রামটি থামবে কি করে? উপরের প্রোগ্রামে যে-সব নির্দেশ দেওয়া আছে তার সাহায্যে প্রোগ্রামটি থামানো সম্ভব নয়। থামাতে হলে কমপিউটারের সুইচ বন্ধ করে দিতে হবে। তবে কি কোন প্রোগ্রামে GO TO নির্দেশটি থাকলে সেই প্রোগ্রাম থামানোর জন্য সব সময়েই কমপিউটারের সুইচ বন্ধ করেই থামাতে হবে? না তা নয়। সাধারণত এই নির্দেশটি অপর একটি নির্দেশের সঙ্গেই দেওয়া হয়। সেই নির্দেশ এরপরেই আলোচনা করা হবে।

এখানে উল্লেখ করা প্রয়োজন যে, কমপিউটারের 40 সংখ্যক নির্দেশ পালন করার অর্থ, তা পরবর্তী 50 সংখ্যক নির্দেশ পালন না করে আবার 10 সংখ্যক নির্দেশ এবং তারপরের নির্দেশগুলি করবে।

IF–THEN-নির্দেশঃ

GO TO নির্দেশের সাহায্যে যেমন যে কোনো একটি লাইন সংখ্যায় সরাসরি যাওয়া যায়, এই নির্দেশের সাহায্যেও তা করা সম্ভব। তবে GO TO নির্দেশ অপর একটি লাইন সংখ্যায় যায় বিনাশর্তে, কিন্তু এক্ষেত্রে তা নয়। এই নির্দেশ একটি শর্তের উপর নির্ভর করে অন্য একটি নির্দেশে যাবে। বারবার কিছু সংখ্যক নির্দেশ করার জন্য IF-THEN এবং GO TO নির্দেশের সাহায্য নেওয়া যেতে পারে। এবং বারবার এই কিছু সংখ্যক নির্দেশ করাকে 'লুপ' বা আবর্ত বলা হয়। IF-THEN নির্দেশটি লেখার নিয়মঃ

1) লাইন সংখ্যা IF শর্ত THEN লাইন সংখ্যা অথবা বেসিকের কোনো নির্দেশ অথবা

2) লাইন সংখ্যা IF শর্ত GO TO লাইন সংখ্যা

নিচের উদাহরণের সাহায্যে বোঝানো যেতে পারে।

উদাহরণ 1

```
100 IF A<B THEN 40
```

এখানে A<B একটি শর্ত। এই নির্দেশের অর্থ হল A, B-র থেকে ছোট হলে অর্থাৎ শর্তটি সত্য হলে 40 সংখ্যক লাইন সংখ্যায় গিয়ে

সেখানকার নির্দেশ পালন করবে। অন্যথায় 100 সংখ্যক লাইনের পরের লাইন সংখ্যার নির্দেশে চলে যাবে।

উদাহরণ 2.

```
100 IF A<B THEN C = A
```

এবারে A, B-র থেকে ছোট হলে A র মান C তে রাখবে এবং তারপর 100 সংখ্যক লাইনের পরের লাইনের সংখ্যার নির্দেশ করবে। A, B র থেকে ছোট না হলে A র মান C-তে না রেখেই পরের লাইন সংখ্যার নির্দেশ করবে।

উদাহরণ 3.

```
100 IF A<B GO TO 40
```

এক্ষেত্রে উদাহরণ 1-এর মতই কাজ হবে।

এখানে THEN বা GO TO-র পরে যে লাইন সংখ্যার উল্লেখ করা আছে তা ওই প্রোগ্রামেরই কোনো একটি নির্দেশের লাইন সংখ্যা হবে এবং তা IF-নির্দেশের আগে বা পরে যে কোনো স্থানেই হওয়া সম্ভব। অবশ্য যদি পরে হয় তবে তা ঠিক পরের নির্দেশটির লাইন সংখ্যা না হয়ে অন্য কোন নির্দেশের লাইন সংখ্যা হওয়া বাঞ্ছনীয়। THEN-এর পর লাইন সংখ্যা না হয়ে বেসিক ভাষার অন্য যে কোনো নির্দেশ হওয়াও সম্ভব। তবে অন্য নির্দেশ হলে তা সাধারণত LET-নির্দেশ হয়। এবারে IF-THEN নির্দেশের সাহায্যে একটি উদাহরণ করে দেখানো হবে। এই উদাহরণটি GO TO নির্দেশের সাহায্যে করে দেখানো হয়েছে। সেখানে দেখা গেছে কমপিউটার বারবার দুটি সংখ্যা পড়ে, যোগ করে, যোগফল ছাপিয়ে আবার দুটি সংখ্যা পড়বে। কখনোই থামবে না। IF-THEN নির্দেশের সাহায্যে এই থামানোর ব্যবস্থা করা যেতে পারে। এটা নানাভাবে করা সম্ভব। একটা উপায়ে কতবার দুটি করে সংখ্যা পড়বে বলে দেওয়া যায়। মনে করা যাক, পাঁচবার পড়বে। সেক্ষেত্রে উদাহরণটি কীভাবে লেখা যেতে পারে, নীচের উদাহরণে দেখানো হয়েছে।

উদাহরণ 4.

```
10 N = 0
20 INPUT A, B
30 C = A + B
40 PRINT C
50 N = N + 1
60 IF N<5 THEN 20
70 END
```

প্রথমবার 10 সংখ্যক লাইনের নির্দেশ থেকে শুরু করে 50 সংখ্যক লাইনের নির্দেশ পালন করার পরে কমপিউটার যখন 60 সংখ্যক নির্দেশে আসবে তখন N-এর মান হচ্ছে 1। এবারে IF-এর পরে যে শর্ত আছে সেই অনুযায়ী N-সর মান 5 থেকে কম হওয়ায় শর্তটি সত্য হবে। কাজেই THEN-এর পরে যে লাইন সংখ্যা দেওয়া আছে সেখানে গিয়ে তার নির্দেশ পালন করবে। এক্ষেত্রে ওই লাইন সংখ্যাটি হল 20। 20 সংখ্যক লাইনে এসে আবার দুটি সংখ্যা পড়ে, যোগ করে, যোগফল ছাপাবার পরে 50 সংখ্যক লাইনের নির্দেশ অনুযায়ী N-এর মান হবে 2। 60 সংখ্যক লাইনে এসে এবারেও শর্তটি সত্য হওয়ায় আবার 20 সংখ্যক লাইনে যাবে। এইভাবে চলতে থাকার পর এক সময়ে N-এর মান 5 হবে অর্থাৎ 5 বার দুটি সংখ্যা পড়ে, যোগ করে, যোগফল ছাপাবার পরে তা 5 হবে। এবারে 60 সংখ্যক লাইনের নির্দেশ পালন করতে গিয়ে দেখা যাবে শর্তটি আর সত্য নেই। কারণ 5 কখনোই 5-এর থেকে ছোট নয়। কাজেই এক্ষেত্রে 60-সংখ্যক লাইনের পরের লাইনে END নির্দেশটি পাওয়াতে থেমে যাবে।

একথা মনে রাখা দরকার যে, IF শব্দটির পর সব সময়েই একটি শর্ত থাকবে। এই শর্ত সম্পর্ক-সূচক চিহ্নের সাহায্যে করা হয়। দুটি সংখ্যা বা নামের মত দুটি একই ধরনের জিনিস তুলনামূলক ভাবে বিচার করতে এই সম্পর্ক-সূচক চিহ্নের প্রয়োজন। নীচে এই সম্পর্ক-সূচক চিহ্নগুলি দেখানো হচ্ছে।

চিহ্ন	সম্পর্ক
=	সমান
<>	অসমান
<	ছোট
>	বড়
<=	ছোট বা সমান
>=	বড় বা সমান

এই চিহ্নগুলির সাহায্যে কি ভাবে শর্ত তৈরি করা যায়? IF N<5 THEN 20। এখানে শর্ত হচ্ছে N-এর মান 5 থেকে ছোট হলে THEN শব্দটির পরে যে লাইন সংখ্যা আছে সেখানে যাবে। আর ছোট না হলে এই নির্দেশের পরের লাইন সংখ্যার নির্দেশ পালন করবে।

অনেক সময়ে দুটি রাশির বদলে দুটি রাশিমালার মধ্যে সম্পর্কের জন্যও এই চিহ্নগুলি ব্যবহার করা হয়। যেমন,

উদাহরণ 5.

```
30 IF A + B <C – D/2 THEN 90
```

এক্ষেত্রে শর্তটি কি ? এখানে প্রথমে A এবং B-এর যোগফল বের করতে হবে। এরপর D-এর মানকে 2 দিয়ে ভাগ করে যে ভাগফল পাওয়া যাবে তা C-এর মান থেকে বাদ দিয়ে বিয়োগফল পাওয়া যাবে। এখন শর্তটি হবে, প্রথম যোগফলটি শেষের বিয়োগফল থেকে ছোট না বড় বা সমান।

একটি শর্তে পাটীগণিতের চিহ্ন (অর্থাৎ যোগ, বিয়োগ, গুণ, ভাগ ইত্যাদি) এবং সম্পর্ক চিহ্ন উভয়েই থাকতে পারে। সেক্ষেত্রে সম্পর্ক চিহ্নের দুপাশের পাটীগণিতের কাজ আলাদাভাবে প্রথমে করতে হবে। এরপর শর্তটি সত্য না মিথ্যা দেখা দরকার।

এবারে প্রবাহ চিত্রের সাহায্যে যে-সব উদাহরণ দেখানো হয়েছিল তার কয়েকটি IF-THEN-এর সাহায্য নিয়ে বেসিক ভাষায় করে দেখানো হবে।

উদাহরণ 6.

দুটি সংখ্যার গরিষ্ঠ সাধারণ গুণনীয়ক বের করো।

সমাধান

```
10 INPUT A%, B%
20 C% = B%/A% 'C%-BEING AN INTEGER NO WILL HAVE
25 'QUOTIENT AS INTEGER
30 C% = B% – C% * A% 'NOW C% WILL HAVE REMAINDER
40 IF C% = 0 THEN 80
50 B% = A%
60 A% = C%
70 GO TO 20
80 PRINT "THE GCD NO. IS"; A%
90 END
```

মনে করা যাক, সংখ্যা দুটি 14 এবং 35। অর্থাৎ A% এবং B%-তে যথাক্রমে 14 এবং 35 থাকবে। এরপর 20 সংখ্যক লাইনের নির্দেশ অনুসারে C%-তে 2 থাকবে। কারণ C% চলরাশির নামের শেষে শতকরা চিহ্ন থাকায় এখানে কেবলমাত্র একটি অখন্ড সংখ্যা রাখা যাবে। 30 সংখ্যক লাইনের নির্দেশ পালন করে C%-এর নূতন মান হবে 35 – 28 অর্থাৎ 7। প্রশ্ন করা যেতে পারে, C% নামটি আগেই একবার ব্যবহার করা হয়েছে, আবার ওই একই নাম ব্যবহার না করে অন্য কোনো নাম লিখলে আপত্তি কোথায় ? না, আপত্তির কিছু নেই। তবে অনাবশ্যক অন্য একটি চলরাশির নাম লেখার জন্য স্মৃতিকোষে আবার আর একটি জায়গা দরকার হবে। তা ছাড়া C% নামের চলরাশিতে সঞ্চিত ভাগফল আর পরে কোনো কাজে লাগবে না। কাজেই এই একই জায়গাতে ভাগশেষও রাখা যেতে

পারে। এরপর 40 সংখ্যক লাইনের নির্দেশে ভাগশেষ শূন্য কিনা পরীক্ষা করে দেখা হচ্ছে। যদি শূন্য হয় তাহলে সরাসরি 80 সংখ্যক লাইনে এসে সেখানকার নির্দেশ অনুযায়ী A%-এর সঞ্চিত মান ছাপাবে এবং এটিই হবে দুটি সংখ্যার গরিষ্ঠ সাধারণ গুণনীয়ক। কিন্তু C%-এর মান শূন্য না হলে কি করা হবে ? এই উদাহরণে প্রথমবার C% শূন্য নয়। কাজেই 50 সংখ্যক লাইনে এসে A%-এর মান B%-তে রাখা হবে। অর্থাৎ B%-তে 14 থাকবে। এরপর A%-তে C%-এর মান অর্থাৎ 7 থাকবে। এবারে 70 সংখ্যক লাইনের নির্দেশ পালন করার অর্থ, সরাসরি 20 সংখ্যক লাইনে চলে যাওয়া। এই লাইনের নির্দেশ পালন করার পর C%-তে 2 থাকবে এবং 30 সংখ্যক লাইনের নির্দেশ অনুযায়ী C%-তে এবারে শূন্য হবে। এরপর 40 সংখ্যক লাইনে এসে C%-তে শূন্য থাকায় সরাসরি 80 সংখ্যক লাইনে গিয়ে A%-এর মান 7 ছাপাবার পর থেমে যাবে।

উদাহরণ 7.

প্রবাহ চিত্র অধ্যায়ের উদাহরণ 6 বেসিক ভাষায় নীচে লেখা হচ্ছে।

```
10 INPUT A
20 IF A = 0 GO TO 120
30 IF A <= 500 THEN 90
40 IF A <= 2000 THEN 70
50 R = .05 * A
60 GO TO 100
70 R = .04 * A
80 GO TO 100
90 R = .02 * A
100 PRINT A, R
110 GO TO 10
120 END
```

সিস্টেম কম্যাণ্ড

সিস্টেম কম্যাণ্ড ঃ

কমপিউটারে সুইচ অন করেই বেসিকে প্রোগ্রাম লেখা শুরু করা যায় না। কমপিউটার চালানোর সঙ্গে সঙ্গেই এর স্মৃতি ভাণ্ডারে কোনো প্রকার ত্রুটি আছে কিনা কমপিউটার প্রথমে তা পরীক্ষা করে দেখে। এই সময়ে ভিডিইউতে RAM TEST শব্দ দুটি ফুটে ওঠে। এই পরীক্ষা শেষ হলে প্রথমে তারিখ এবং তারপরে সময় জানতে চাওয়া হয়। সঠিক তারিখ জানিয়ে কী-বোর্ডে রিটার্ন বোতাম টিপলে তবেই কমপিউটার সময় জানতে চায়। এবারেও সময় জানিয়ে রিটার্ন বোতাম টিপলে ভিডিইউতে 'ডস' প্রম্পট্ ফুটে ওঠে। 'ডস' প্রম্পট্ হল A, B বা C-এর পর '>' চিহ্ন। যেমন,

A>   B>   বা   C>

A, B বা Cর মধ্যে কোন অক্ষর লিখে '>' চিহ্ন দেবে তা নির্ভর করে ডিসক্ অপারেটিং সিস্টেম (DOS-Disk Operating System) প্রোগ্রামগুলি কোন নামের ড্রাইভে আছে তার উপরে। একথা এখানে বলে রাখা ভাল যে, কমপিউটার তারিখ এবং সময় জানতে চাইলে কিছু না জানিয়ে রিটার্ন বোতাম টিপলেও ওই একই 'ডস' প্রম্প্‌টি ফুটে উঠবে। এরপর বেসিকের যে নামের ইন্টারপ্রিটার বা কমপাইলার-ওই মেশিনে আছে তার নাম কী-বোর্ডের মাধ্যমে দিতে হবে। BASICA নামের ইন্টারপ্রিটার থাকলে লিখতে হবে—

A>BASICA

এর ফলে কমপিউটার নীচের লাইনগুলি ভিডিইউতে দেখাবে।

The ABC Personal Computer BASIC

Version 1. 13

(C) Copyright ABC Computer Corp, 1983

(C) Copyright Microsoft 1982

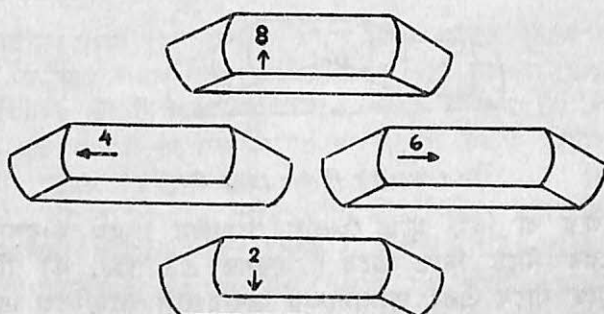
61818 Bytes free
Ok

1LIST 2RUN 3LOAD"4SAVE" 5CONT 6, "LPT1 7TRON 8TROFF 9KEY OSCREEN

এখানে যে সংস্থা এই ইন্টারপ্রিটার লিখেছে একদম প্রথম লাইনটিতে সেই সংস্থার নাম থাকে। এই নাম অবশ্যই এক এক কমপিউটারে এক একরকম হওয়া সম্ভব। এবারে দ্বিতীয় লাইনটি সম্বন্ধে আলোচনা করা প্রয়োজন। প্রথমে যখন কোনো ইন্টারপ্রিটার বা কমপাইলার বাজারে বেরোয় তখন তাতে কিছু ত্রুটি থাকা স্বাভাবিক। কিছু ত্রুটি একত্রে সংশোধন করে আবার ওই নামেই নতুন একটি ইন্টারপ্রিটার বা কমপাইলার-এর সংস্করণ বাজারে বোরোয়। সঙ্গে সঙ্গে এটি ইন্টারপ্রিটারের কত-তম সংস্করণ তা দ্বিতীয় লাইন থেকে দেখা যায়। এরপর কিছু ফাঁকা জায়গার পর 4টি লাইন থাকে। এখানের তৃতীয় লাইন থেকে বোঝা যায় ওই কমপিউটারে বেসিকের প্রোগ্রামের জন্য কতটা ফাঁকা জায়গা আছে। এরপর OK শব্দ লেখা থাকে, OK শব্দটি এবং শেষ লাইনের মধ্যে পর্দায় অনেকটা জায়গা ফাঁকা পাওয়া যায়। সেখানেই প্রোগ্রামের নির্দেশ দেওয়া শুরু করতে হবে। বেসিকে প্রোগ্রাম লেখার বা চালানোর বা প্রোগ্রামের নির্দেশে কোনো ত্রুটি থাকলে তা পরে সংশোধন করার মত নানা কাজের জন্য কতগুলি নির্দেশ দেওয়া হয়। এই ধরনের নির্দেশকে সিস্টেম কম্যাণ্ড বলে। বিভিন্ন ধরনের সিস্টেম কম্যাণ্ড সম্বন্ধে আলোচনার পূর্বে যে কী-বোর্ডের মাধ্যমে এই কম্যাণ্ড দেওয়া হয় সেই কী-বোর্ড সম্বন্ধে কিছু জানা দরকার।

এই কী-বোর্ড সাধারণত টাইপ রাইটারের মত দেখতে। এর তিনটি অংশ থাকে। একেবারে বাম দিকের অংশে 10টি কী (F1 থেকে F10) থাকে। এদের ফাংশন কী নামে অভিহিত করা হয়। এক এক ভাষায় এই দশটি কী-এর এক একরকম কাজ। তবে বেসিকে এই কী দিয়ে কি কাজ করা হয় তা পর্দায় যে শেষের লাইনটি লেখা থাকে তার থেকেই কিছুটা আন্দাজ করা সম্ভব। যেমন, LIST, এর অর্থ F1-কী টিপলে পর্দায় LIST শব্দটি ফুটে

১৩ সংখ্যক চিত্র : একটি কী-বোর্ড

উঠবে। এরপর রিটার্ন কী ব্যবহার করলে যে প্রোগ্রামের LIST চাওয়া হচ্ছে সেই প্রোগ্রামটির প্রথম নির্দেশ থেকে শেষ নির্দেশ পর্যন্ত সবকটি নির্দেশ পর্দায় ফুটে উঠবে। এই দশটি কী সম্বন্ধে বিস্তারিত আলোচনা পরে করা হবে। টাইপরাইটারে সাধারণত যে কী থাকে কী-বোর্ডের মধ্যিখানের অংশে সেই রকমের কী পাওয়া যায়। ডানদিকের অংশের কীগুলিকে সংখ্যার কী বলে। এই কী-বোর্ডের ছবি নীচে দেখানো হল। এবারে ডান পাশের কীগুলি সম্বন্ধে কিছু আলোচনা করা যাক। প্রথমে Num Lock সম্বন্ধে কিছু বলা দরকার। Numeric Lock শব্দ দুটি থেকে Num Lock এসেছে। এই কী টিপে অন্য কোনো সংখ্যার কী টিপলে সেই সংখ্যা পর্দায় ফুটে ওঠে। কিন্তু Num Lock-এর বদলে অন্য কোনো কী-এর ক্ষেত্রে কিন্তু সেই কী-তে সংখ্যার নীচে যে চিহ্ন থাকবে কমপিউটার সেই চিহ্ন ধরে কাজ করবে। এখানে চারটে কারসার নিয়ন্ত্রণের কী-এর কথাই ধরা যাক। প্রথমে অবশ্য জানা দরকার, কারসার কি ? পর্দায় সব সময়ে একটি বিশেষ চিহ্ন দেখা যায়। এই চিহ্নকেই কারসার বলে চিহ্নিত করা হয়। কোনো অক্ষর বা চিহ্ন কী-বোর্ডের মাধ্যমে টাইপ করলে এই চিহ্ন পর্দায় যে স্থানে থাকে সেই স্থানে ওই অক্ষর বা চিহ্নটি ফুটে ওঠে এবং কারসারটি ওই অক্ষর বা চিহ্নের ডানদিকে সরে যায়। এরপর কোনো অক্ষর টাইপ করলে আবার ওই কারসারের স্থানে অক্ষরটি পর্দায় ফুটে উঠবে এবং কারসার ডানদিকে সরে যাবে। এবারে ওই চারটি কারসার নিয়ন্ত্রণ কী-এর সম্বন্ধে আলোচনা করা যাক।

১৪ সংখ্যক চিত্র : কারসার নিয়ন্ত্রণের কী

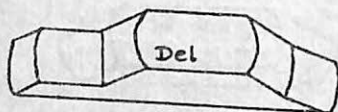
Num Lock না টিপে ডানদিকের কী টিপলে কারসার ডানদিকের একটা অক্ষরের পর সরে যাবে। যতবারই ওই কী টেপা হবে ততবারই কারসার এক ঘর ডানদিকে সরে যেতে থাকবে। এক ঘর বলতে এখানে একটা অক্ষর পর্দায় যে জায়গা নেয় তা বোঝায়।

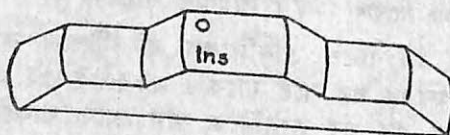
আবার ডানদিকের বদলে বাম দিকের কী টিপলে কারসার এক ঘর করে বামদিকে সরবে। কিন্তু উপরের কী-এর বেলাতে কি হবে? তখন কারসার উপরের দিকে এক লাইন উঠে যাবে। অনুরূপভাবে নীচের কী ব্যবহার করলে কারসার একটি করে নীচের লাইনে নেমে যাবে। অর্থাৎ পর্দায় একটি করে লাইন উপরে উঠে আসবে। কিন্তু Num Lock টেপার পরে এই কী-গুলি টিপলে ওই কী-তে যে সংখ্যা লেখা আছে পর্দায় তা ফুটে উঠবে। সেক্ষেত্রে আর কারসার নিয়ন্ত্রণের কী-গুলি কাজ করবে না।

উদাহরণের সাহায্যে কারসার নিয়ন্ত্রণের কী সম্বন্ধে বোঝানো যেতে পারে । মনে করা যাক, পর্দায় নীচের লাইনটি আছে।

20 A = B + C

এখন ধরা যাক, কারসার 2-এর নীচের আছে, এবং লাইনটিতে B + C-এর স্থানে B – C লিখতে হবে। কিভাবে তা করা যাবে? ডানদিকের কী এর সাহায্যে কারসার 2-এর থেকে সরে 0-এর নীচে আসবে। আবার ওই কী টিপে 0 এবং A-এর মধ্যেকার ফাঁকা জায়গাতে কারসার সরে যাবে। এইভাবে ডানদিকের কী যতবার টেপা হবে ততবারই একঘর করে সরে যাবে। যখন যোগ চিহ্ন (+)-এর নীচে কারসার আসবে তখন বিয়োগ চিহ্নের (–) কী টিপলেই যোগ-এর জায়গাতে বিয়োগ চলে আসবে। এইরকম করে কারসার নিয়ন্ত্রণের কী টিপেই কারসারকে বাম, ডান, উপর ও নীচে করা সম্ভব।

পর্দায় কোনো অক্ষর বা চিহ্ন মুছে ফেলতে চাইলে তাহলে নীচের কী-এর সাহায্যে তা করা সম্ভব।

১৫ সংখ্যক চিত্র : Del-কী

যে অক্ষর বা চিহ্ন মুছে ফেলার প্রয়োজন প্রথমে কারসারটিকে সেই অক্ষরের নীচে নিতে হবে। এরপর এই Del কী টিপলেই অক্ষরটি মুছে যাবে এবং ডানদিকের অক্ষরগুলি বামদিকে এক ঘর করে সরে আসবে।

১৬ সংখ্যক চিত্র : Ins কী

দুটি অক্ষরের মধ্যে এক বা একাধিক অক্ষর ঢোকানোর প্রয়োজন হলে এই কী-এর সাহায্য নিতে হবে। একটি উদাহরণের সাহায্যে এই কী-এর কাজ বোঝানো যেতে পারে। মনে করা যাক, পর্দায় নীচের লাইনটি আছে ঃ

50 A = B – E

এখন আসলে 50 সংখ্যক লাইনটি হওয়ার কথা

50 A = B + C/D – E

অর্থাৎ B এবং '–' চিহ্নের মধ্যে '+C/D' এই চারটি অক্ষর বা চিহ্ন বসাতে হবে। কি ভাবে তা করা হবে এবারে তা দেখা যাক। প্রথমে কারসারটিকে কারসার নিয়ন্ত্রণের কী-এর সাহায্যে '–' চিহ্নের নীচে নিয়ে আসতে হবে। এরপর এই Ins কী টিপে পরপর +, C, /, এবং D চিহ্ন চারটির কী টিপতে হবে। অর্থাৎ প্রথমবার '+' চিহ্ন ব্যবহারের পর '-' চিহ্ন এবং তার ডান দিকের সবকটি অক্ষরই একঘর করে ডান দিকে সরে যাবে। এরপর C লিখলে আবার '–' চিহ্ন এবং তার ডান দিকের অক্ষরগুলি একঘর ডান দিকে সরে যাবে। এভাবে যখনই কোনো একটি অক্ষর ঢোকানো হবে তখনই ডানদিকের সবকটি অক্ষর এক ঘর করে সরে যেতে থাকবে।

একবার ঢোকানো শুরু করলে তা বন্ধ করার জন্য আবার এই কী-টি ব্যবহারের দরকার হবে। এবারে সিস্টেম কম্যাণ্ডের কথায় আসা যাক। বেসিক ভাষায় প্রোগ্রাম লেখার সময়ে এইসব কম্যাণ্ডের প্রয়োজন হয়।

AUTO:

প্রোগ্রাম লেখা শুরু করার আগে এই কম্যাণ্ড দিলে প্রোগ্রামের নির্দেশ দেওয়ার সময় লাইন-সংখ্যা দিতে হয় না। AUTO শব্দটি লিখে রিটার্ন কী টিপলে কমপিউটার পরের লাইনে 10 লিখে থেমে যাবে। এরপর 10 সংখ্যক লাইনের নির্দেশ লিখে রিটার্ন কী-এর সাহায্যে পরের লাইনে 20 লেখা হবে। এইভাবে যে কোনো লাইনের নির্দেশ লেখার পরে রিটার্ন কী-এর সাহায্যেই পরের লাইন-সংখ্যা পর্দায় ফুটে উঠবে। পরের লাইন-সংখ্যা আগের লাইন-সংখ্যার সঙ্গে 10 যোগ করে লেখা হয় বলেই AUTO কম্যাণ্ড নিলে আর লাইন সংখ্যাগুলি দেওয়ার প্রয়োজন হয় না। সেগুলি নিজের থেকেই লেখা হয়ে যায়।

এবারে অন্যান্য সিস্টেম কম্যাণ্ডের কথায় আসায় যাক।

NEW:

একথা আগেই বলা হয়েছে যে, একটি বেসিক ইন্টারপ্রিটারের

নাম লিখে রিটার্ন কী-এর সাহায্যে কমপিউটার কয়েকটি লাইন লেখার পর OK শব্দটি লিখবে। এই OK শব্দটি পাওয়ার পর বেসিক ভাষায় লেখা কোনো প্রোগ্রাম কী-বোর্ডের সাহায্যে লিখে চালানো সম্ভব। প্রথম প্রোগ্রামটি চালানোর পরে আবার অন্য কোনো প্রোগ্রাম চালাতে হলে এবারে NEW কম্যাণ্ড দিতে হবে। এর কাজ আগের প্রোগ্রামটিকে 'র‍্যাম' থেকে মুছে ফেলা। এই নির্দেশটি কার্যকরী করতে হলে কী বোর্ডের সাহায্যে NEW শব্দটি লিখে রিটার্ন কী দিতে হবে। আগের প্রোগ্রামটি 'র‍্যাম' থেকে মোছা হয়ে গেলে ভিডিইউতে OK শব্দটি আসবে। এই শব্দটি আসার পরে আবার আগের মতই নতুন প্রোগ্রামের নির্দেশ দেওয়া যাবে। NEW শব্দটি না লিখে যদি কোনো নির্দেশ দেওয়া হয় তবে তা আগের প্রোগ্রামের নির্দেশ হিসেবেই ধরে নেবে। একটি উদাহরণের সাহায্যে এটি পরিষ্কার করা যেতে পারে।

```
OK
10 DATA 5.2, 3.6
20 READ A, B
30 PRINT A + B
40 END
```

এই প্রোগ্রামটি চালালে কমপিউটার ভিডিইউতে নীচের সংখ্যাটি ছাপিয়ে OK শব্দটি ছাপাবে।

```
8.8
OK
```

এবারে NEW শব্দটি না লিখে যদি নীচের নির্দেশগুলি দেওয়া হয় তা হলে কি হবে ?

```
15 DATA 6.4, 5.2
25 READ A, B
35 PRINT A – B
40 END
```

এবারে প্রোগ্রামটি চালানোর পর কমপিউটার কি ছাপাবে ?

```
11.6
1.2
OK
```

কেন এমন হল ? মনে রাখতে হবে NEW শব্দটি লেখা হয় নি। কাজেই পুরনো প্রোগ্রামের নির্দেশ র‍্যাম থেকে মুছে যায় নি। সেক্ষেত্রে প্রোগ্রামটির চেহারা হবে-

```
10 DATA 5.2, 3.6
15 DATA 6.4, 5.2
20 READ A, B
25 READ A, B
30 PRINT A + B
35 PRINT A – B
40 END
```

এবারে দেখা যাক, প্রোগ্রামটি কিভাবে কাজ করবে ? 20 সংখ্যক লাইনের READ নির্দেশ অনুসারে A এবং B-তে যথাক্রমে 5.2 এবং 3.6 থাকবে। এরপর 25 সংখ্যক লাইনের নির্দেশ অনুযায়ী আবার A ও B-তে যথাক্রমে 6.4 এবং 5.2 থাকবে। কাজেই A এবং B-এর পুরনো মান মুছে যাবে। এরপর 30 সংখ্যক লাইনে PRINT নির্দেশ পালন করে কমপিউটার 11.6 ছাপাবে এবং 35 সংখ্যক লাইনের PRINT নির্দেশ অনুসারে 1.2 ছাপানো হবে। কিন্তু NEW শব্দটি লিখে রিটার্ন কী চালানোর পরে যদি এই নতুন চারটি লাইনের নির্দেশ দেওয়া হত তাহলে প্রোগ্রামটি চালানো হলে কেবলমাত্র 1.2 ছাপিয়ে OK শব্দটি লিখে কমপিউটার থেমে যেত। আসলে পুরনো প্রোগ্রামের নির্দেশগুলি আর কমপিউটারের স্মৃতিতে না থাকায় যে নতুন নির্দেশগুলি দেওয়া হবে কেবলমাত্র সে নির্দেশগুলি অনুসারেই কাজ হবে।

LIST ঃ

স্মৃতিতে বেসিকে লেখা যে প্রোগ্রামটি সঞ্চয় করা আছে তা ভিডিইউ-এর পর্দায় ফুটিয়ে তোলার জন্য এই নির্দেশ ব্যবহার করা হয়। এই নির্দেশের কিছু রকমফের আছে। নীচে বিভিন্ন রকমের LIST-এর ব্যবহার দেখানো হচ্ছে।

শুধু LIST শব্দটি লিখে রিটার্ন কী টিপলে কমপিউটার সমস্ত প্রোগ্রামটাই স্মৃতি থেকে নিয়ে এসে ভিডিইউ-এর পর্দায় লিখবে।

অনেক সময়ে কেবলমাত্র একটি লাইন দেখার প্রয়োজন হতে পারে। সেক্ষেত্রে LIST শব্দটি লিখে যে সংখ্যক লাইনের প্রয়োজন সেই সংখ্যাটি লেখা হয় এবং এরপর রিটার্ন কী-এর সাহায্যে ওই লাইনটি পর্দায় ফুটে উঠবে। যেমন, LIST 70। এবারে 70 সংখ্যক লাইনটি পর্দায় লেখা হয়ে যাবে।

পরপর কতগুলি লাইন দেখার প্রয়োজন হলে LIST শব্দটি লিখে যে সংখ্যার লাইন থেকে আরম্ভ করে যে সংখ্যার লাইন পর্যন্ত দেখা দরকার সেই সংখ্যা দুটি লিখে সংখ্যা দুটিকে পৃথক করার জন্য

মধ্যে একটি বিয়োগ চিহ্ন দেওয়া হয়। যেমন, LIST 100-150 লিখে রিটার্ন কী ব্যবহার করলে প্রোগ্রামটির 100 সংখ্যক লাইন থেকে শুরু করে 150 সংখ্যক লাইন পর্যন্ত পর্দায় ফুটে উঠবে। অনেক সময়ে, প্রোগ্রামের কোনো একটি লাইন থেকে শুরু করে পরের সবকটি লাইনই পর্দায় নিয়ে আসার প্রয়োজন। যেমন, 100 সংখ্যক লাইন থেকে শুরু করে প্রোগ্রামের শেষের সবকটি দেখতে হলে লিখতে হবে LIST 100- । এর পর রিটার্ন কী-এর সাহায্যেই কাজ হবে। প্রোগ্রামে মোট কতগুলি নির্দেশ আছে তা জানা না থাকলে এই নির্দেশের দ্বারা তা করা সম্ভব। আবার অনেক সময়ে প্রথম থেকে শুরু করে মাঝের কোনো লাইন পর্যন্ত দেখার প্রয়োজন হতে পারে। যদি 70 সংখ্যক লাইন পর্যন্ত দেখার দরকার হয় তাহলে নির্দেশ দিতে হবে LIST –70 এবং এরপর রিটার্ন কী ব্যবহার করতে হবে।

LIST ব্যবহার করার আগে মনে রাখা দরকার যে, স্মৃতিতে নিশ্চই এর আগে কোনো প্রোগ্রাম লেখা হয়েছে। কিন্তু স্মৃতিতে কোনো প্রোগ্রাম যদি না থাকে অথচ LIST ব্যবহার করা হয়, তাহলে পর্দায় কিছুই ফুটে উঠবে না।

স্মৃতিতে সঞ্চয় করা আছে এমন কোনো প্রোগ্রাম পর্দায় দেখতে হলে যেমন LIST ব্যবহার করা হয় তেমনি কমপিউটারের সঙ্গে লাগানো প্রিন্টারে ছাপানোর প্রয়োজন হলে LLIST লিখতে হবে।

LIST শব্দটি দুভাবে লেখা সম্ভব। একঃ L, I, S, T এই চারটি অক্ষর কী বোর্ডের সাহায্যে লেখা; দুইঃ ফাংশন কী F1-এর সাহায্যেই এই LIST শব্দটি পর্দায় ফুটে উঠবে। অর্থ্যাৎ FI কী টিপলেই পর্দায় LIST শব্দটি ফুটে উঠবে।

EDIT ঃ

অনেক সময়ে একটি সমস্যা সমাধান করার জন্য প্রথমে যেসব নির্দেশ দেওয়া হয় তার মধ্যে কয়েকটি লাইনের নির্দেশে ভুল ত্রুটি থেকে যাওয়া সম্ভব। এই ত্রুটি সংশোধনের জন্যে EDIT নির্দেশটি ব্যবহার করা হয়। যে সংখ্যক লাইনে ভুল আছে EDIT শব্দটির পরে সেই সংখ্যাটি লিখে রিটার্ন কী ব্যবহার করলেই পর্দায় ওই লাইনটি ফুটে উঠবে। উদাহরণ হিসেবে যদি EDIT 40 লিখে রিটার্ন কী ব্যবহার করা হয়, তাহলে 40 সংখ্যক লাইনটি পর্দায় লেখা হবে। এরপরে এই লাইনের যে যে জায়গাতে পরিবর্তনের প্রয়োজন সেইসব স্থানে কারসারকে একে একে নিয়ে যেতে হবে। কারসারকে যথাস্থানে নিয়ে গিয়ে সঠিক অক্ষর বা চিহ্নটি ব্যবহার করলেই তা পুরনো অক্ষর বা চিহ্নের জায়গাতে লেখা হয়ে যাবে।

DELETE ঃ

কমপিউটারের স্মৃতিতে যে বেসিক প্রোগ্রাম সঞ্চয় করা আছে সেই প্রোগ্রামের এক বা একাধিক লাইন বাদ দেওয়ার প্রয়োজনে এই নির্দেশটি ব্যবহার করা হয়। এই নির্দেশটিতে DELETE শব্দের পর যে লাইন সংখ্যা লেখা থাকবে সেই লাইনটি প্রোগ্রাম থেকে বাদ যাবে। যেমন, DELETE 60 লিখে রিটার্ন কী ব্যবহার করলে প্রোগ্রামের 60 সংখ্যক লাইনটি প্রোগ্রাম থেকে বাদ যাবে। আবার DELETE 60–90 লিখে রিটার্ন কি ব্যবহার করলে প্রোগ্রাম থেকে 60 থেকে 90 পর্যন্ত সবকটি লাইনই বাদ যাবে। কিন্তু DELETE 60– লেখা হলে 60 সংখ্যক লাইন থেকে শুরু করে শেষ লাইন পর্যন্ত সব লাইনই মুছে যাবে। আবার DELETE –60 লিখলে প্রথম থেকে শুরু করে 60 সংখ্যক লাইনের নির্দেশগুলি মুছে যাবে। এখানে একটা কথা মনে রাখা দরকার যে, DELETE-এ দুটি লাইন সংখ্যা থাকলে বাম দিকের লাইন সংখ্যা সব সময়েই ডান দিকের লাইন সংখ্যা থেকে কম হওয়া প্রয়োজন। নাহলে DELETE-এর কাজ হবে না অর্থাৎ লাইন মুছবে না।

RUN ঃ

একটি বেসিকের প্রোগ্রাম লিখে EDIT কম্যাণ্ডের সাহায্যে নির্ভুল করার পরে তা চালানোর উপযুক্ত হয়। এই প্রোগ্রাম চালিয়ে সমাধান পাওয়ার জন্য RUN কম্যাণ্ডের প্রয়োজন। RUN শব্দটি পর্দায় লিখে রিটার্ন কী ব্যবহার করলে কমপিউটার প্রোগ্রামটির নির্দেশগুলি পালন করতে শুরু করে।

RUN শব্দটিকে কী-বোর্ডের সাহায্যে R, U এবং N এই তিনটি অক্ষর আলাদা ভাবে না লিখে ফাংশান কী F2-এর সাহায্যেও করা সম্ভব। F2 ব্যবহারে পর্দায় RUN শব্দটি ফুটে উঠবে। মনে রাখতে হবে, RUN-এর বেলায় F2 ব্যবহার করলে আর রিটার্ন কী-এর প্রয়োজন হয় না।

SAVE ঃ

এমন অনেক প্রোগ্রাম লেখা হয় যার প্রয়োজন একবার চালালেই শেষ হয়ে যায় না। ভবিষ্যতে বেশ কয়েকবার এই প্রোগ্রাম চালানোর দরকার হতে পারে। সেক্ষেত্রে যখনই ওই প্রোগ্রাম চালানোর প্রয়োজন হবে তখন আবার নতুন করে প্রোগ্রামের নির্দেশগুলি না লিখেও তা করা যেতে পারে। প্রথমবারই প্রোগ্রামের নির্দেশগুলি ডিসকে সঞ্চয় করা গেলে এমনটি সম্ভব। ডিস্কে প্রোগ্রাম সঞ্চয় করার জন্য এই SAVE নির্দেশটির প্রয়োজন। এই নির্দেশ

লেখার নিয়ম কি ? দৃষ্টান্তস্বরূপ নেওয়া যাক –

SAVE "TEST1"

SAVE শব্দটি লেখার পরে দুটি উদ্ধৃতি চিহ্নের মধ্যে একটি নাম লেখা হয়। এরপর রিটার্ন কী-এর সাহায্যেই ওই নামে প্রোগ্রামটি ডিস্কে সঞ্চিত করে রাখা হয়। এরপর যখনই প্রয়োজন হবে TESTI নামের সাহায্যে প্রোগ্রামটিকে ডিস্ক থেকে কমপিউটারের স্মৃতিতে নিয়ে আসা যাবে। এই প্রোগ্রাম নাম আটটি অক্ষর পর্যন্ত হওয়া সম্ভব। কেবলমাত্র কোলন (:) চিহ্ন ছাড়া আর যে কোনো অক্ষরই নাম হিসেবে ব্যবহার করা যায়। অনেক সময়ে যে ড্রাইভটিতে কাজ চলছে প্রোগ্রামটি সেখানে সঞ্চয় না করে অন্য কোনো ড্রাইভে সঞ্চয় করার প্রয়োজন হতে পারে। সেক্ষেত্রে নির্দেশটি কেমন হবে ? দৃষ্টান্তস্বরূপ নেওয়া যাক–

SAVE "B:TEST1"

এখানে ধরে নেওয়া হচ্ছে যে, A ড্রাইভে কাজ চলছিল এবং TEST1 নামের প্রোগ্রামটি B ড্রাইভের ফ্লপি ডিস্কে সঞ্চয় করে রাখা হল। এখানে প্রশ্ন হতে পারে, এবারে কোলন চিহ্ন ব্যবহার সম্ভব হল কি করে ? মনে রাখতে হবে, আগে বলা হয়েছিল নামে কোলন চিহ্ন ব্যবহার করা যায় না। এখানে নাম কিন্তু কেবলমাত্র TEST1 এবং B:-এর সাহায্যে বোঝানো হচ্ছে B-ড্রাইভ। এই SAVE শব্দটি এবং প্রথম উদ্ধৃতি চিহ্নটি কী-বোর্ডে S, A, V, E এবং " এই পাঁচটি অক্ষর বা চিহ্ন আলাদা ভাবে না ব্যবহার করে কেবলমাত্র ফাংশন কী F4 দ্বারাই করা সম্ভব।

LOAD:

এই নির্দেশের সাহায্যে যে বেসিক প্রোগ্রাম ফ্লপি ডিসকে সঞ্চয় করা আছে সেই প্রোগ্রাম আবার কমপিউটারের স্মৃতিতে নিয়ে আসা যায়। প্রোগ্রামটি TEST1 নামে A-ড্রাইভের ফ্লপি ডিসকে থাকলে এবং এখন A-ড্রাইভেই কাজ চললে এই নির্দেশটি লেখা হবে

LOAD "TEST1"

এই নির্দেশ লেখার পর রিটার্ন কী ব্যবহার করলেই TEST1 প্রোগ্রামটি আবার স্মৃতিতে চলে আসবে। ফাংশন কী F4-এর সাহায্যে যেমন SAVE এবং একটি উদ্ধৃতি চিহ্ন (") পর্দায় ফুটে ওঠে, F3 কী দিয়েও তেমনি LOAD শব্দটি এবং একটি উদ্ধৃতি চিহ্ন পর্দায় ফুটে উঠবে। এরপর যে প্রোগ্রামটিকে স্মৃতিতে আনতে হবে সেই প্রোগ্রামের নাম লিখে আবার একটি উদ্ধৃতি চিহ্ন দিয়ে রিটার্ন কী-এর সাহায্যে ওই প্রোগ্রামটি স্মৃতিতে এসে যাবে।

TRACE:

অনেক সময়ে একটি সমস্যা সমাধানের জন্য কমপিউটারে যে নির্দেশ দেওয়া হয় তা প্রথম বারেই একেবারে নির্ভুল নাও হতে পারে। নানা কারণে তা হওয়া সম্ভব। যিনি প্রোগ্রাম লিখছেন তিনি হয়তো সমস্যাটি বুঝতে ভুল করলেন। অথবা সমস্যাটি ঠিকমতো বুঝেও নির্দেশ দেওয়ার সময়ে অসাবধানতাবশত কিছু ভুল নির্দেশ দিতে পারেন। এর ফলে সমস্যাটির সমাধান ঠিকমত পাওয়া যাবে না। একটি বড় প্রোগ্রামে ভুল বের করা সহজ নয়। TRACE কম্যাণ্ডের সাহায্যে এই ভুল বের করা সহজ হবে। এই কম্যাণ্ডটি হল TRON অর্থাৎ TRACE ON। ফাংশন কী F7 ব্যবহার করলেই এই TRON শব্দটি পর্দায় ফুটে উঠবে। এবারে দেখা যাক, এই কম্যাণ্ড প্রোগ্রামে কোথায় ব্যবহার করা হবে এবং তা কি ভাবে কাজ করবে। প্রথমে একটি প্রোগ্রাম লেখা যাক।

```
10 I = 1
20 A = 2 * I – 1
30 PRINT A
40 I = I + 1
50 IF I<4 THEN 20
60 END
```

এর ঠিক পরের লাইনেই TRON শব্দটি লিখে রিটার্ন কী ব্যবহার করতে হবে। ফলে OK শব্দটি পর্দায় লেখা হবে। এরপর RUN শব্দটি লিখে রিটার্ন কী ব্যবহার করলেই পর্দায় নীচের লেখা ফুটে উঠবে।

```
[10] [20] [30] 1
[40] [50] [20] [30] 3
[40] [50] [20] [30] 5
[40] [50] [60]
```

তৃতীয় বন্ধনীর মধ্যেকার সংখ্যাগুলি হল লাইন সংখ্যা। এর সাহায্যে বোঝা যাবে, প্রোগ্রামটি কোন লাইনের পরে কোন লাইনের নির্দেশ পালন করছে। প্রথম লাইনের সংখ্যাগুলি দেখে বোঝা যাচ্ছে 10, 20 এবং 30 সংখ্যক লাইনের নির্দেশ পরপর করছে। এরপরের 1 সংখ্যাটি কি ? 30 সংখ্যক লাইনে PRINT নির্দেশে যে চলরাশির নাম উল্লেখ আছে 1 হল তার মান। এরপর 40, 50 সংখ্যক লাইনের নির্দেশ করে আবার 20, 30 সংখ্যক লাইনের নির্দেশ করছে। কমপিউটার নির্দেশগুলি ঠিকভাবে পালন করছে কিনা অর্থাৎ যে নির্দেশের পর যে নির্দেশটি করার কথা তা

ঠিকমত করছে কিনা এবং যে-সব মানগুলি ছাপা হচ্ছে তা ঠিক কিনা এর থেকে তা বোঝা যাবে। অনেক সময়ে প্রোগ্রামে ভুল বের করার জন্য প্রোগ্রামের মাঝে মাঝে ইচ্ছেমত কিছু PRINT নির্দেশ দেওয়া হয় এবং সেই PRINT নির্দেশে প্রয়োজনীয় কয়েকটি চলরাশির নামের উল্লেখ থাকে। এরপর TRON-এর সাহায্যে দেখে নেওয়া হয় ওইসব চলরাশির মান ঠিকমত আসছে কিনা। প্রোগ্রামটি ঠিক হয়ে গেলে ওইসব PRINT নির্দেশগুলি মুছে ফেলতে হবে। তা DELETE নির্দেশের সাহায্যে করা যায়।

এই পদ্ধতিকে কমপিউটারের ভাষায় ডিবাগিং (Debugging) বলে। ডিবাগিং-এর অর্থ, প্রোগ্রামে যে বাগ (bug) অর্থাৎ ভুল আছে তা খুঁজে বের করা।

TRON-এর কাজ শেষ হবার পরে অর্থাৎ প্রোগ্রামটি নির্ভুল হলে TROFF শব্দটি লিখে রিটার্ন কী ব্যবহার অর্থাৎ TRACE OFF করতে হবে। না হলে প্রোগ্রাম চললেই কমপিউটার আবার আগের মতই তৃতীয় বন্ধনীর মধ্যে লাইনের সংখ্যাগুলি লিখবে এবং যেসব জায়গাতে PRINT নির্দেশ আছে সেখানে ওইসব চলরাশির মান ছাপাবে। TROFF শব্দটি ফাংশন কী F8-এর দ্বারাই লেখা হয়ে যায়। মনে রাখতে হবে, F7 বা F8 কী ব্যবহার করলে আর রিটার্ন কী ব্যবহারের প্রয়োজন নেই।

CONT:

একটি প্রোগ্রামের চলা অবস্থায় ছেদ ঘটিয়ে আবার ঠিক তারপর থেকে শুরু করার জন্য এই নির্দেশের প্রয়োজন। অনেক সময়ে কয়েকটি নির্দেশ বেশ কিছু সংখ্যক বার হয়তো 20000 বা তারও বেশি সংখ্যক বার করা দরকার। সেক্ষেত্রে কমপিউটারের অনেক সময় লেগে যাবে। কিছুক্ষণ কমপিউটার চলার পরে দেখে নেওয়া সম্ভব কতদূর পর্যন্ত কাজ এগিয়েছে। দেখার পর CONT কম্যান্ডের সাহায্যে আবার যে পর্যন্ত কাজ করা হয়েছে তারপর থেকে শুরু করা সম্ভব হবে। দৃষ্টান্তস্বরূপ একটি উদাহরণ নেওয়া যাক।

```
100 I = 1
110 J = 2*I - 1
130 I = I + 1
140 IF I <= 20000 THEN 110
```

এরপর RUN লিখে প্রোগ্রামটি চালানো হল। কিছুক্ষণ চলার পরে ctrl এবং C কী দুটির সাহায্যে পর্দায় নীচের অক্ষর দুটি ফুটে ওঠবে।

```
^C
```

এবং এরপর কমপিউটার নিজে থেকেই নীচের লাইন দুটি ছাপাবে

```
Break in 110 (বা 130 বা 140)
OK
```

I-এর কোন মান পর্যন্ত এতক্ষণ করা হয়েছে তা দেখার জন্য নির্দেশ দেওয়া যেতে পারে

```
PRINT I
```

এর ফলে 1575 ছাপালে বুঝতে হবে I-এর মান 1575 পর্যন্ত করা হয়েছে। কমপিউটার 1575 ছাপানোর পরে আবার OK লিখবে। এরপর CONT এই কম্যাণ্ডটি দিয়ে রিটার্ন কী ব্যবহার করলেই কমপিউটার আবার 1575-এর পর থেকে কাজ করতে আরম্ভ করবে। সুতরাং প্রয়োজন মত Ctrl এবং C-এই কী দুটির সাহায্যে চালু প্রোগ্রামকে থামানো সম্ভব এবং তারপর আবার সেই জায়গা থেকে আরম্ভ করার জন্য CONT লিখে রিটার্ন কী ব্যবহার করতে হবে। CONT শব্দটি আলাদা করে না লিখে ফাংশন কী F5-এর সাহায্যে ওই শব্দটি লেখা হয়ে যাবে এবং সেক্ষেত্রে আর রিটার্ন কী ব্যবহারের দরকার নেই। CONT অক্ষর চারটি CONTINUE শব্দ থেকে নেওয়া হয়েছে।

KEYঃ

বেসিকে দশটি ফাংশন কী ব্যবহার করা সম্ভব। KEY LIST শব্দ দুটি লিখে রিটার্ন কী-এর দ্বারা পর্দায় কি ফুটে উঠবে তা নীচে দেখানো হল।

ফাংশন কী	পর্দায় লেখা হবে
F1	LIST
F2	RUN←
F3	LOAD"
F4	SAVE"
F5	CONT←
F6	,"LPTI:"←
F7	TRON←
F8	TROFF←
F9	KEY
F10	SCREEN 0, 0, 0←

একথা এখানে বলে রাখা ভাল যে, F2, F5, F6, F7, F8, F10 কীগুলির সাহায্যে পর্দায় লেখাগুলি ফুটে উঠবে এবং এরপর আর রিটার্ন কী ব্যবহারের প্রয়োজন হবে না। যেমন, F2-এর দ্বারা পর্দায় RUN লেখা হবে এবং সঙ্গে সঙ্গে প্রোগ্রামের নির্দেশগুলিও পালন করা শুরু হয়ে যাবে। এরপর আর আলাদা করে রিটার্ন কী ব্যবহারের প্রয়োজন নেই।

অনেক সময়েই সবগুলি ফাংশন কী ব্যবহার হয় না, কাজেই যে ফাংশন কী ব্যবহার করলে পর্দায় যে লেখা ফুটে ওঠে এবং তার জন্য যে কাজ হয় তার পরিবর্তে অন্য কোনো কাজ করার জন্য ওই ফাংশন কী-কে ব্যবহার করা সম্ভব। এই কাজ করার জন্য KEY কম্যাণ্ডের প্রয়োজন। উদাহরণ হিসেবে উল্লেখ করা যাক –

```
10 KEY 6, "NEW"
```

এর পর যখনই ফাংশন কী F6 ব্যবহার করা হবে তখনই NEW শব্দটি পর্দায় ফুটে উঠবে। কাজেই সেক্ষেত্রে আর আলাদাভাবে N, E, W অক্ষর তিনটির কী ব্যবহার করার দরকার হবে না। F6 কী যা সাধারণত করার কথা তা না করে এবার থেকে ওই NEW শব্দটি পর্দায় লিখবে। এতে সুবিধে কি ? এক প্রোগ্রামের পর অন্য একটি বেসিকের প্রোগ্রাম পর্দায় লেখার আগে NEW লিখতে হয়। এবারে যখনই নতুন প্রোগ্রাম পর্দায় লেখার দরকার হবে তখনই ওই F6 কীর সাহায্যেই লেখা যাবে।

প্রোগ্রামে বারবার কিছু সংখ্যক নির্দেশ পালন করার নির্দেশ

কমপিউটারে অনেক সমস্যা সমাধানের সময়ে বারবার কিছু সংখ্যক নির্দেশ পালন করার প্রয়োজন হয়। একে কমপিউটারের ভাষায় লুপ বা আবর্ত বলা হয়। এই ধরনের কাজ করার জন্য IF-THEN নির্দেশের ব্যবহার আগেই দেখানো হয়েছে। এই নির্দেশ ছাড়াও আরও কিছু সংখ্যক নির্দেশ এই একই ধরনের কাজের জন্য বেসিকে ব্যবহার করা হয়ে থাকে। সেইসব নির্দেশ এবারে একে একে আলোচনা করা হবে।

IF-THEN-ELSE - নির্দেশঃ

IF-THEN নির্দেশের বেলাতে IF-এর পর যে শর্তটি থাকে তা সত্য হলে কমপিউটার THEN-এর পরের নির্দেশ পালন করে। তা না হলে IF-THEN-এর পরের লাইনে যে নির্দেশ থাকে তা পালন করে থাকে। কিন্তু IF-THEN-ELSE-এর ক্ষেত্রে IF-এর পরের শর্তটি সত্য না হলে ELSE শব্দের পর যে নির্দেশ থাকবে কমপিউটার তা পালন করবে। ELSE-এর পর কেবলমাত্র একটি লাইন সংখ্যা থাকতে পারে অথবা একটি IF-THEN নির্দেশ কিংবা অপর একটি IF-THEN-ELSE থাকা সম্ভব। অবশ্য লাইন সংখ্যা থাকলে IF শব্দের পরের শর্তটি অসত্য হলে ওই লাইন সংখ্যায় গিয়ে কমপিউটার সেখানকার নির্দেশ পালন করবে। কিন্তু IF-THEN বা IF-THEN-ELSE থাকলে আগের মতই পরের কোন নির্দেশ পালন করা হবে, তা এই নতুন IF শব্দের পরের শর্তটির উপর নির্ভর করবে। এবারে IF-THEN-ELSE-এর ছকটি নীচে দেখানো হচ্ছে।

IF শর্ত THEN এক বা একাধিক নির্দেশ
ELSE এক বা একাধিক নির্দেশ

তবে THEN এবং ELSE-এর পর একাধিক নির্দেশ থাকলে নির্দেশগুলিকে আলাদা করার জন্য দুটি নির্দেশের মধ্যে কোলন (:) চিহ্ন ব্যবহার করা হয়। একটা উদাহরণ নেওয়া যাক।

উদাহরণ 1

```
80 IF A<60 THEN  T = .2*E : I = 1
               ELSE  T =  0 : I = 2
```

উপরের উদাহরণে A, 60-এর থেকে ছোট হলে E-এর মানকে .2 দিয়ে গুণ করে T-তে রাখা হবে এবং তারপর I-এর মান 1 হবে। কিন্তু A, 60-এর থেকে ছোট না হলে ELSE-এর পর যে দুটি নির্দেশ আছে, অর্থাৎ T = 0 এবং I = 2, কমপিউটার সেই কাজ করবে। এখানে উল্লেখ করা প্রয়োজন যে, 80 সংখ্যক লাইনে IF নির্দেশ শুরু হয়েছে। সেইজন্য কমপিউটারের সঙ্গে যে ভিডিইউ থাকে সেখানে ELSE শব্দ এবং তার পরের নির্দেশ দুটি পরের লাইনে লেখা হলেও এগুলিও 80 সংখ্যক লাইনের নির্দেশ হিসেবেই ধরা হবে।

এবারে প্রবাহ চিত্র অধ্যায়ের উদাহরণ-5 IF-THEN-ELSE-এর সাহায্যে লিখে দেখানো হবে।

উদাহরণ 2.

```
10 REM THE PROGRAM FINDS THE GCD OF TWO NUMBERS
20 REM THE NUMBERS ARE A% AND B%
30 REM A% IS SMALLER OR EQUAL TO B%
40 INPUT A%, B%
50 D% = B%/A%
60 C% = B% - A% * D%
70 REM C% IS THE REMAINDER OF B%/A%
80 IF C% = 0 THEN PRINT A% : GO TO 90
        ELSE B% = A% : A% = C% : GO TO 50
90 END
```

উপরের উদাহরণে নামের শেষে শতকরা চিহ্ন (%) ব্যবহার করে বোঝানো হচ্ছে যে, এইসব চলরাশির মান কেবলমাত্র অখণ্ড সংখ্যা হতে পারে। মনে করা যাক, A%-তে 14 এবং B%-তে 35 সংখ্যা দুটি রাখা হল। 50 সংখ্যক লাইনের নির্দেশ অনুসরণ করে 35-কে 14 দিয়ে ভাগ করে ভাগফল 2 D%-তে রাখা হবে। এরপর 60

সংখ্যক লাইনে 14-কে 2 দিয়ে গুণ করে 35 থেকে বাদ-দিয়ে ভাগশেষ 7 পাওয়া যাবে এবং এই 7 সংখ্যাটিই হল C% নামের চলরাশির মান। এবার 80 সংখ্যক লাইনে C%-এর মান শূন্য কিনা তা পরীক্ষা করে দেখা হচ্ছে। এক্ষেত্রে শূন্য না হওয়ায় ELSE শব্দের পর যে-সব নির্দেশ আছে কমপিউটার তা পালন করবে। অর্থাৎ এবারে B%-এর নতুন মান হবে 14 এবং A%-এর 7 এবং এরপর GO TO 50 নির্দেশের জন্য আবার 50 সংখ্যক লাইনে এসে B%-কে A%-এর মান দিয়ে ভাগ করে D%-এর মান হবে 2। এরপর 60 সংখ্যক লাইনের নির্দেশ অনুসরণ করে C%-এর মান হবে শূন্য। কাজেই 80 সংখ্যক লাইনের নির্দেশ পালন করার সময়ে এবারে শর্তটি সত্য হওয়ায় A%-এর মান 7 ছাপানো হবে এবং তারপর কমপিউটার থামবে। এই 7-ই হচ্ছে 14 এবং 35 সংখ্যা দুটির গরিষ্ঠ সাধারণ গুণনীয়ক। এখানে উল্লেখ করা যেতে পারে যে, 60 এবং 80 সংখ্যক লাইনের মধ্যেকার 70 সংখ্যক লাইনের নির্দেশটি কমপিউটার কখনোই পালন করছে না। এই নির্দেশে REM শব্দটি থাকায় কমপিউটার সব সময়েই 60-এর পরেই 80 সংখ্যক লাইনের নির্দেশ পালন করবে। কেবলমাত্র প্রোগ্রামটি ছাপানো হলে তবেই ওই 70 সংখ্যক লাইনটিও ছাপানো হবে। আবার 80 সংখ্যক লাইনে একাধিক নির্দেশ থাকলেও তার জন্য একটি লাইন সংখ্যা থাকবে এবং এটি একটি নির্দেশ হিসেবেই ধরা হবে। তবে একথা মনে রাখা দরকার যে, এখানে প্রথম লাইনের শেষ অক্ষর P-এর পরে রিটার্ন কী টিপতে হবে না। প্রথম লাইনের শেষ অক্ষরটি লেখার পর ফাঁকা জায়গা লেখার জন্য যে কী তা টিপে যেতে হবে। এর ফলে নিজের থেকেই ভিডিইউতে পরের লাইনে চলে যাবে। দ্বিতীয় লাইনে যা লেখা প্রয়োজন তা কী-বোর্ডের মাধ্যমে লিখে যেতে হবে।

প্রবাহ চিত্র অধ্যায়ের উদাহরণ 6 এবারে IF-THEN-ELSE নির্দেশের সাহায্যে কিভাবে লেখা যাবে তা দেখানো যাক।

উদাহরণ 3.

```
10 REM THIS PROGRAM CALCULATES REBATE
20 REM THE REBATE DEPENDS ON THE SALE VALUE
30 INPUT "SALE VALUE =", A
40 IF A = 0 THEN 80
50 IF A < = 500 THEN R = .02 * A
        ELSE IF A < = 2000 THEN R = .04 * A
        ELSE R = .05 * A
60 PRINT "SALE VALUE =" ; A , "REBATE =" ; R
```

```
70 GO TO 30
80 END
```

এখানে 50 সংখ্যক লাইনের নির্দেশ পালন করার সময়ে A-এর মান 500 সংখ্যাটির থেকে ছোট বা তার সমান কিনা প্রথমে দেখতে হবে। যদি তা হয় তবে R-এর মান হবে A-এর 2%। অন্যথায় আবার A-এর মান 2000-এর থেকে ছোট বা সমান কিনা পরীক্ষা করে দেখতে হবে। এবারে শর্তটি সত্য হলে A-কে .04 দিয়ে গুণ করে R-এর মান পাওয়া যাবে। সত্য না হলে R হবে A-এর 5%। প্রথম শর্ত সত্য হলে R-এর মান বের করে কমপিউটার একেবারে 60 সংখ্যক লাইনে চলে যাবে। সত্য না হলেই পরের শর্তটি পরীক্ষা করে দেখা হবে। এরপর ছাপানো নির্দেশটি পালন করে আবার 30 সংখ্যক লাইনে এসে কমপিউটার সেখানকার নির্দেশ অনুসরণ করবে। অর্থাৎ A-এর অপর একটি মান কী-বোর্ডের মাধ্যমে দেওয়া হবে। এইভাবে একের পর এক A-এর মানের জন্য R-এর মান বের করে ছাপানো হবে। প্রোগ্রামটিকে থামানোর জন্য শেষবার A-এর মান শূন্য দেওয়া হবে। এবারে 40 সংখ্যক লাইনে এসে সেখানকার নির্দেশ পালন করার অর্থ হবে থেমে যাওয়া। প্রোগ্রামটিকে থামানোর জন্য 40 সংখ্যক লাইনের নির্দেশ প্রয়োজন।

FOR-NEXT - নির্দেশঃ

এর পূর্বে লুপ বা আবর্তের জন্য IF THEN এবং IF-THEN-ELSE-নির্দেশের ব্যবহার হয়েছে। এবারে ওই একই ধরনের কাজের জন্য FOR-NEXT নির্দেশ দুটি ব্যবহার করে দেখানো হবে। পাঁচটি ভিন্ন ভিন্ন সংখ্যার গড় বের করার সমস্যার সমাধানে এই নির্দেশ ব্যবহার করে দেখানো হচ্ছে।

উদাহরণ 1.

```
10 SUM = 0
20 FOR N = 1 TO 5 STEP 1
30 INPUT A
40 SUM = SUM + A
50 NEXT N
60 AV = SUM/5
70 PRINT AV
80 END
```

এখানে 20 সংখ্যক লাইনের FOR নির্দেশ কিভাবে কাজ করছে ? প্রথমে N চলরাশির মান 1 ধরা হবে এবং ওই মান TO শব্দের পর

যে সংখ্যা আছে তার থেকে বড় কিনা অর্থাৎ এক্ষেত্রে 5-এর থেকে বড় কি না পরীক্ষা করে দেখতে হবে। যদি বড় না হয় তা হলে কমপিউটার পরের নির্দেশগুলি করবে। কিন্তু পরের কোন নির্দেশ পর্যন্ত ? তা বোঝানোর জন্য NEXT নির্দেশের প্রয়োজন। উপরের 30 সংখ্যক লাইনের নির্দেশ থেকে শুরু করে 50 সংখ্যক লাইনের নির্দেশ পর্যন্ত কমপিউটার করবে। NEXT নির্দেশ পালন করার অর্থ আবার 20 সংখ্যক লাইনে এসে N-এর বর্তমান মান-এর সঙ্গে STEP শব্দের পর যে সংখ্যা আছে তা যোগ করা। এখানে STEP-এর পর 1 থাকায় দ্বিতীয়বার N-এর মান 2 পাওয়া যাবে। এরপর এই মান আবার 5-এর সঙ্গে পরীক্ষা করে দেখা হবে। 5-এর থেকে বড় না হলে আবার 30 সংখ্যক লাইনের নির্দেশ থেকে আরম্ভ করে কমপিউটার 50 সংখ্যক লাইনের নির্দেশ পর্যন্ত করবে। এইভাবে চলতে থাকবে। যে মুহূর্তে N-এর মান 5 থেকে বড় হবে তখনই NEXT নির্দেশ যে লাইনে আছে তার পরের লাইনে গিয়ে সেখানকার নির্দেশ পালন করবে। এখানে 60 সংখ্যক লাইনে এসে AV-এর মান বের করে পরের লাইনের নির্দেশ অনুসারে তা ছাপাব এবং এর পরের নির্দেশে END থাকায় প্রোগ্রামটি থেমে যাবে।

উপরের উদাহরণ থেকে FOR এবং NEXT নির্দেশ দুটি কি ভাবে কাজ করে তা বোঝা গেল। এবারে FOR নির্দেশের ধরন সম্বন্ধে বলা যেতে পারে।

FOR চলরাশি = প্রাথমিক মান TO চূড়ান্ত মান STEP সংযোজন মান

এই নির্দেশ FOR শব্দ দিয়ে শুরু হয়। এই শব্দের পর একটি চলরাশির নাম এবং তারপর সমান চিহ্ন থাকে। সমান চিহ্নের ডান পাশে যে সংখ্যা থাকে তাকে চলরাশির প্রাথমিক মান হিসেবে ধরা হয়। এরপর TO শব্দ এবং তারপরের সংখ্যাটি হল চূড়ান্ত মান অর্থাৎ চলরাশির মান ওই পর্যন্ত হওয়া সম্ভব। চূড়ান্ত মানের পর STEP শব্দ এবং তারপর যে সংখ্যাটি থাকে এর পর কমপিউটার যতবার FOR নির্দেশ পালন করবে ততবারই সেই সংখ্যাটি চলরাশির তখনকার বর্তমান মানের সঙ্গে যোগ করা হবে। এইভাবে কয়েকবার যোগ করার পর চলরাশির মান চূড়ান্ত মানের চেয়ে বড় হবে। সেক্ষেত্রে NEXT-এর পরের লাইনের নির্দেশে কমপিউটার সরাসরি চলে যাবে।

এবারে FOR-NEXT-এর সম্বন্ধে বিস্তারিত আলোচনা করা হবে।

1. ডান পাশের মানগুলি বিভিন্ন ধরনের সংখ্যা হতে পারে।

   i) পূর্ণ সংখ্যা

   ii) ভগ্নাংশ

একটি উদাহরণের সাহায্যে ভগ্নাংশের ব্যবহার দেখানো যাক। x-এর বিভিন্ন মানের জন্য নীচের সূত্র অনুসারে y-এর মান বের করতে হবে।

$y = x^2$

x-এর বিভিন্ন মান হল 1, 1.1, . . . , 2। নীচে প্রোগ্রামটি দেখানো হচ্ছে।

উদাহরণ 2.

```
10 FOR X = 1 TO 2 STEP .1
20 Y = X * X
30 PRINT X, Y
40 NEXT X
50 END
```

iii) ঋণাত্মক সংখ্যা

উদাহরণ 3.

```
10 FOR X = –5 TO 5 STEP 1
```

উদাহরণ 4.

```
10 FOR X = 10 TO 1 STEP –1
```

এর পূর্বে বলা হয়েছে, চলরাশির মান যখন চূড়ান্ত মানের থেকে বড় হবে তখন কমপিউটার NEXT-নির্দেশের পরের লাইনের নির্দেশ করবে। কিন্তু STEP-এর পর সংখ্যাটি ঋণাত্মক হলে কমপিউটার ঠিক তার বিপরীত কাজই করবে। অর্থাৎ যতক্ষণ পর্যন্ত না চলরাশির মান চূড়ান্ত মানের থেকে ছোট হচ্ছে ততক্ষণ FOR নির্দেশ থেকে শুরু করে NEXT নির্দেশ পর্যন্ত নির্দেশগুলি বারবার করতে থাকবে। আর যে মুহূর্তে ছোট হবে তখনই NEXT-নির্দেশের পরের লাইনের নির্দেশ পালন করবে।

2. TO এর আগে, পরে এবং STEP-এর পর সংখ্যা না হয়ে চলরাশির নাম থাকাও সম্ভব।

উদাহরণ 5.

```
50 FOR X = A TO B STEP C
```

তবে এক্ষেত্রে 50 সংখ্যক লাইনের নির্দেশ পালন করার পূর্বে A, B এবং C চলরাশিগুলির মান অবশ্যই কমপিউটারকে জানতে হবে। না হলে 50 সংখ্যক লাইনের নির্দেশ পালন করা সম্ভব হবে না।

3. STEP শব্দের পর সংখ্যাটি 1 হলে STEP শব্দটি এবং সংখ্যাটি লেখা না থাকলেও কমপিউটার STEP-এর পর 1 ধরে নিয়ে কাজ করবে।

উদাহরণ 6.

```
10 FOR N = 1 TO 5
```

এখানে STEP এবং তার পরে কোনো সংখ্যার উল্লেখ না থাকা সত্ত্বেও প্রথমবারের পর কমপিউটার যখনই FOR নির্দেশ পালন করবে তখনই N-এর বর্তমান মানের সঙ্গে 1 যোগ করা হবে।

4. FOR নির্দেশে সমান চিহ্নের বাম পার্শ্বে যে চলরাশির নাম থাকে NEXT নির্দেশেও সেই একই চলরাশির নামের উল্লেখ থাকা প্রয়োজন।

5. একটি FOR নির্দেশের শুরু এবং সেই FOR নির্দেশের জন্য যে NEXT নির্দেশ থাকে তার মধ্যে অপর একটি FOR নির্দেশ এবং তার NEXT নির্দেশ থাকা সম্ভব। একটি দৃষ্টান্ত নেওয়া যাক। মনে করা যাক, নীচের সূত্রে ভিন্ন ভিন্ন N এবং R-এর মানের জন্য A-এর মান বের করতে হবে।

এখানে $A = 1000 (1 + R/100)^N$.

ধরা যাক, N-এর মান 1, 2, 3, 4, 5 হতে পারে এবং প্রতিটি N-এর মানের জন্য R = 10, 15, 20, 25 হওয়া সম্ভব। নীচে বেসিক প্রোগ্রামটি লক্ষ্য করি।

উদাহরণ 7.

```
10 FOR N = 1 TO 5
20 FOR R = 10 TO 25 STEP 5
30 A = 1000 * (1 + R/100) ^ N
40 PRINT N, R, A
50 NEXT R
60 NEXT N
70 END
```

এখানে প্রথমে N-এর মান 1 দিয়ে শুরু হবে। এই মান 5-এর থেকে বড় না হওয়ায় 20 সংখ্যক লাইনে এসে সেখানকার নির্দেশ অনুসারে R-এর মান হল 10। এরপর 30 এবং 40 সংখ্যক লাইনের নির্দেশ পালন করে 50 সংখ্যক লাইনে NEXT নির্দেশে R থাকায় কমপিউটার আবার 20 সংখ্যক লাইনে ফিরে আসবে। এবারে R-এর মানের সঙ্গে 5 যোগ করে R-এর বর্তমান মান হবে 15 এবং তা 25-এর থেকে বড় না হওয়ায় আবার 30 থেকে 50 সংখ্যক লাইনের নির্দেশ পালন করবে। N-এর মান কিন্তু সেই 1-ই থাকবে। এরপর আবার 20-তে এসে R-এর মানের সঙ্গে 5 যোগ করা হবে। এইভাবে এক সময়ে R-এর মান 25-এর থেকে বড় হবে এবং তখন 50 সংখ্যক লাইনের পরের লাইনে NEXT নির্দেশে N চলরাশির

নাম থাকায় এবারে আবার 10 সংখ্যক লাইনে এসে N-এর মান 1 বাড়িয়ে 2 হবে এবং তা 5-এর থেকে বড় না হওয়ায় আবার 20 সংখ্যক লাইনে এসে R-এর মান 10 দিয়ে শুরু করবে। এইভাবে N-এর প্রতিটি মানের জন্য R, 10 থেকে শুরু করে 25 পর্যন্ত হবে। এইভাবে এক সময়ে R, 25-এর থেকে বড় এবং N, 5-এর থেকে বড় হবে। এই অবস্থায় NEXT N-এর পরের লাইনে এসে থেমে যাবে।

6. একটি FOR-NEXT-এর মধ্যে অপর একটি FOR নির্দেশ শুরু হলে এই দ্বিতীয় FOR-এর NEXT নির্দেশটি প্রথম NEXT নির্দেশের আগেই থাকবে। উদাহরণ 6-এ 10 সংখ্যক লাইনে FOR নির্দেশ এবং ওই FOR-এর NEXT নির্দেশ 60 সংখ্যক লাইনে দেওয়া হয়েছে। অপর একটি FOR 20 সংখ্যক লাইনে থাকায় ওই FOR-এর জন্য NEXT নির্দেশ 60 সংখ্যক লাইনের আগেই থাকতে হবে। এক্ষেত্রে তা 50 সংখ্যক লাইনে রয়েছে। নীচের চিত্রটি দেখা যাক।

```
┌--10 FOR N = 1 TO 5
¦ ┌20 FOR R = 10 TO 25 STEP 5
¦ ¦        .
¦ ¦        .
¦ ¦        .
¦ └50 NEXT R
└--60 NEXT N
```

তবে এরকম না হয়ে নীচের মত হলে কিন্তু ভুল হবে।

```
┌--10 FOR N = 1 TO 5
¦ ┌20 FOR R = 10 TO 25 STEP 5
¦ ¦        .
¦ ¦        .
¦ ¦        .
└-¦-50 NEXT N
  └60 NEXT R
```

7. নীচের চিত্রের মত হওয়াও সম্ভব।

```
┌--10 FOR I = 1 TO 10
¦ ┌20 FOR J = 5 TO 10
¦ ¦        .
¦ ¦        .
¦ ¦        .
¦ └50 NEXT J
¦
```

```
┌─60 FOR K = 10 TO 20 STEP 2
│┌-70 FOR L = .1 TO -.1 STEP -0.1
││      .
││      .
││      .
│└-100 NEXT L
└--110 NEXT K
----120 NEXT I
```

একটি আবর্তের মধ্যে অন্য একটি আবর্ত থাকলে তাকে কমপিউটারের ভাষায় নেস্টেড লুপ বলা হয়।

FOR-NEXT নির্দেশের সাহায্যে এবারে কোনো সংখ্যার ফ্যাক্টোরিয়াল বের করে দেখানো যাক। ফ্যাক্টোরিয়াল কি ? N সংখ্যাটির ফ্যাক্টোরিয়াল হবে

$n! = n \times (n - 1) \times (n - 2) \times . . . \times 1,$

অর্থাৎ $5! = 5 \times 4 \times 3 \times 2 \times 1 = 120.$

উদাহরণ 8. 2 থেকে আরম্ভ করে 20 পর্যন্ত সংখ্যার ফ্যাক্টোরিয়াল নীচে দেওয়া হল।

```
10 F =1
20 FOR I=2 TO 20
30 F= F*I
40 LPRINT "the number is = ";I, " the factorial is = ";F
50 NEXT I
60 END
```

উপরের প্রোগ্রামটি কি ভাবে কাজ করবে ? প্রথমে F-এর মান 1 হবে। এরপর 20 সংখ্যক লাইনের নির্দেশ অনুসারে I এর মান 2 হবে এবং তা 20 থেকে ছোট হওয়ায় পরের লাইন সংখ্যার নির্দেশ অনুসারে F-এর পুরনো মান 1-এর সঙ্গে I-এর মান 2 গুণ-করে F-এর নতুন মান হবে 2। এরপর LPRINT নির্দেশটি পালন করে আবার 20 সংখ্যক লাইনে এসে I-এর মান 3 হবে এবং এবারেও এই মান 20 থেকে ছোট থাকায় পরের লাইনের নির্দেশ পালন করে F-এর নতুন মান হবে 6। এইভাবে ফ্যাক্টোরিয়াল 2, 3, 4 ইত্যাদির জন্য মান বের করে ছাপানো হবে। I-এর মান যতক্ষণ পর্যন্ত 20 থেকে ছোট বা সমান থাকবে ততক্ষণ কমপিউটার এই কাজ করবে। আবার যে মুহূর্তে I-এর মান 21 হবে তখনই NEXT I নির্দেশের পরের লাইনে এসে প্রোগ্রামটি থেমে যাবে।

এই প্রোগ্রামটি কমপিউটারে চালিয়ে যে ফলাফল পাওয়া গেছে তা দেখা যাক।

```
the number is = 2     the factorial is = 2
the number is = 3     the factorial is = 6
the number is = 4     the factorial is = 24
the number is = 5     the factorial is = 120
the number is = 6     the factorial is = 720
the number is = 7     the factorial is = 5040
the number is = 8     the factorial is = 40320
the number is = 9     the factorial is = 362880
the number is = 10    the factorial is = 3628800
the number is = 11    the factorial is = 3.99168E+07
the number is = 12    the factorial is = 4.790016E+08
the number is = 13    the factorial is = 6.227021E+09
the number is = 14    the factorial is = 8.717829E+10
the number is = 15    the factorial is = 1.307674E+12
the number is = 16    the factorial is = 2.092279E+13
the number is = 17    the factorial is = 3.556874E+14
the number is = 18    the factorial is = 6.402374E+15
the number is = 19    the factorial is = 1.216451E+17
the number is = 20    the factorial is = 2.432902E+18
```

উপরের ফলাফল লক্ষ্য করলে বুঝতে অসুবিধে হয় না যে, 12 সংখ্যা পর্যন্ত ফ্যাক্টোরিয়াল ঠিকই হয়েছে, কিন্তু তারপর থেকে একটু ভুল এসে যাচ্ছে। কেন এমন হল ? বেসিক নিয়ে আলোচনা শুরু করার সময়েই বলা হয়েছিল, যে-সব সংখ্যা একক-দৈর্ঘ্য তারা 6 অঙ্ক পর্যন্ত নিখুঁত হয়। উপরের চলরাশি F-এ একক-দৈর্ঘ্য সংখ্যা রাখা সম্ভব। কাজেই 12 সংখ্যা পর্যন্ত সঠিক পাওয়া গেছে। কিন্তু 20 সংখ্যা পর্যন্ত সঠিক পেতে হলে কি করতে হবে ? চলরাশির নাম এমন হওয়া প্রয়োজন যেখানে দ্বি-দৈর্ঘ্য সংখ্যা রাখা সম্ভব। নামের শেষে সংখ্যা চিহ্ন (#) দিয়ে বোঝানো হয়ে থাকে যে এই চলরাশিতে দ্বি-দৈর্ঘ্য সংখ্যা রাখা যায়। নীচের প্রোগ্রামটি চালিয়ে যে ফলাফল পাওয়া গেছে তা প্রোগ্রামটির পরই দেওয়া হল।

```
10 F# = 1
20 FOR I=2 TO 20
30 F# = F#*I
```

```
40 LPRINT "the number is = ";I, " the factorial is = ";F#
50 NEXT I
60 END
```

```
the number is = 2     the factorial is = 2
the number is = 3     the factorial is = 6
the number is = 4     the factorial is = 24
the number is = 5     the factorial is = 120
the number is = 6     the factorial is = 720
the number is = 7     the factorial is = 5040
the number is = 8     the factorial is = 40320
the number is = 9     the factorial is = 362880
the number is = 10    the factorial is = 3628800
the number is = 11    the factorial is = 39916800
the number is = 12    the factorial is = 479001600
the number is = 13    the factorial is = 6227020800
the number is = 14    the factorial is = 87178291200
the number is = 15    the factorial is = 1307674368000
the number is = 16    the factorial is = 20922789888000
the number is = 17    the factorial is = 355687428096000
the number is = 18    the factorial is = 6402373705728000
the number is = 19    the factorial is = 1.21645100408832D+17
the number is = 20    the factorial is = 2.43290200817664D+18
```

উপরের প্রোগ্রামে F-এর পর সংখ্যা চিহ্ন (#) দিয়ে বোঝানো হচ্ছে, এই চলরাশির মান দ্বি-দৈর্ঘ্য সংখ্যা হতে পারে। এবারে ফলাফলে কিন্তু কোনো রকম ভুল হবে না। এক্ষেত্রে 16 অঙ্ক পর্যন্ত নিখুঁত পাওয়া যায়। কেন 16 তা দ্বি-দৈর্ঘ্য সংখ্যা আলোচনা করার সময়েই বলা হয়েছে। কাজেই 20 সংখ্যা পর্যন্ত ফ্যাক্টোরিয়াল সঠিক পাওয়া যাবে। কিন্তু যে-সব চলরাশির মান কেবলমাত্র অখণ্ড সংখ্যা হয় সেরকম চলরাশি ব্যবহার করলে কি ফল পাওয়া যাবে তা দেখতে হলে নামের ক্ষেত্রে শতকরা চিহ্ন (%) ব্যবহার করে দেখতে হবে। এই ধরনের প্রোগ্রাম এবং সেই প্রোগ্রাম কি ফল দেখাবে তা নীচে দেওয়া হল।

```
10 F%=1
20 FOR I=2 TO 20
30 F% =F%*I
```

```
40 PRINT "the number is = " ; I, " the factorial is
 = " ; F% "
50 NEXT I
60 END
OK
RUN
the number is = 2   the factorial is = 2
the number is = 3   the factorial is = 6
the number is = 4   the factorial is = 24
the number is = 5   the factorial is = 120
the number is = 6   the factorial is = 720
the number is = 7   the factorial is = 5040
Overflow in 30
OK
```

প্রোগ্রামটি চালানোর পর দেখা গেল 7-এর ফ্যাক্টোরিয়াল বের করার পর Overflow in 30 লাইনটি লিখে কমপিউটার থেমে গেল। আমরা আগেই আলোচনা করেছি যে, সবচেয়ে বড় অখণ্ড সংখ্যা 32767 হতে পারে। এখানে 8-এর ফ্যাক্টোরিয়াল হবে 5040 × 8 অর্থাৎ 40320। কিন্তু এই অখণ্ড সংখ্যাটি F%-এ রাখা সম্ভব নয়। কাজেই যে মুহূর্তে 32767-এর থেকে বড় সংখ্যা কমপিউটার F%-এ তে রাখার চেষ্টা করতে গেছে তখনই overflow হয়েছে। 30 এই সংখ্যা দিয়ে বোঝানো হচ্ছে যে এই সংখ্যার লাইনের নির্দেশ পালন করার সময়ে Overflow হয়েছে।

WHILE-WIND-নির্দেশঃ

WHILE-WEND নির্দেশের সাহায্যেও কিছু সংখ্যক নির্দেশ বারবার করা সম্ভব। WHILE নির্দেশ লেখার ধরন নীচে দেখানো হল।

WHILE শর্ত

এখানে যতক্ষণ পর্যন্ত WHILE শব্দের পরের শর্তটি সত্য থাকছে ততক্ষণ কমপিউটার WHILE নির্দেশ থেকে শুরু করে WEND নির্দেশ পর্যন্ত যে সব নির্দেশ থাকবে তা করবে। যে মুহূর্তে শর্তটি অসত্য হবে তখনই WEND নির্দেশের পরের লাইনে নিয়ন্ত্রণ চলে যাবে। নীচের উদাহরণে WHILE-WEND-এর ব্যবহার দেখানো হল।

উদাহরণ 1.

```
10 SUM = 0
20 N = 1
30 WHILE N < = 5
40 INPUT A
50 SUM = SUM + A
60 N = N + 1
70 WEND
80 AV = SUM/5
90 PRINT AV
100 END
```

FOR নির্দেশে চলরাশির প্রাথমিক মান এবং FOR নির্দেশে আবার নিয়ন্ত্রণ ফিরে এলে যে মান যোগ করে চলরাশির নতুন মান তৈরি হয় তারও উল্লেখ থাকে। কিন্তু WHILE-এর ক্ষেত্রে যে চলরাশির মানের উপর নির্ভর করে নির্দেশগুলি বারবার করা হবে সেই চলরাশির প্রাথমিক মান WHILE নির্দেশের আগেই প্রোগ্রামে দিতে হয়। এরপর যে মান একবার নির্দেশগুলি করার পর চলরাশির মানের সঙ্গে যোগ করা হয় তা WHILE নির্দেশের পর কিন্তু WEND নির্দেশের আগে প্রোগ্রামে থাকা প্রয়োজন। সেইজন্য কোনো কোনো ক্ষেত্রে FOR-NEXT-এর বদলে WHILE-WEND ব্যবহার করলে প্রোগ্রামে দুটি বেশি লাইন সংখ্যার প্রয়োজন হয়। তবে এমন অনেক সমস্যা আছে যেখানে কিছু সংখ্যক নির্দেশ কতবার করা হবে তা জানা নেই। সেইসব ক্ষেত্রে একটি শর্ত দেওয়া থাকে এবং শুধুমাত্র এটুকু জানা থাকে যে, যতক্ষণ পর্যন্ত ওই শর্তটি সত্য থাকবে ততক্ষণ ওই নির্দেশগুলি করে যেতে হবে। আবার শর্তটি মিথ্যে হলে ওই নির্দেশগুলি না করে কমপিউটার WEND-এর পরে যে লাইন সংখ্যা থাকবে সেখানে গিয়ে ওই লাইন সংখ্যার নির্দেশ করবে। একটি উদাহরণের সাহায্যে তা বোঝানো যাক।

উদাহরণ 2. প্রবাহ চিত্র অধ্যায়ের উদাহরণ 6 নীচে WHILE-WEND দিয়ে দেখানো হচ্ছে।

```
10 INPUT A
20 WHILE A <> O
30 IF A ∠= 500 THEN R = .02*A ELSE IF A<=2000
THEN R =.04*A ELSE R =.05*A
40 LPRINT A,R
50 INPUT A
```

```
60 WEND
70 END
```

A	R
3500	175
2500	125
1500	60
1000	40
400	8
100	2

এখানে 10 সংখ্যক লাইনের নির্দেশে একটি ক্রয়মূল্য পড়া হচ্ছে এবং তার জন্য কমিশন বের করার আগেই ওই ক্রয়মূল্য শূন্য কিনা দেখে নেওয়া দরকার। যদি শূন্য হয় তবে সরাসরি 70 সংখ্যক লাইনের নির্দেশ অনুসারে প্রোগ্রামটি থেমে যাবে। কিন্তু শূন্য না হলে এই ক্রয়মূল্যের জন্য কমিশন বের করে ছাপিয়ে আবার অপর একটি ক্রয়মূল্য পড়া হবে আর পড়ার পরই তা শূন্য কিনা 20 সংখ্যক লাইনে এসে দেখা হবে। এখানে কত সংখ্যক ক্রয়মূল্য পড়া হবে সে সম্বন্ধে সঠিক তথ্য জানা না থাকায় এই 20 সংখ্যক লাইনের নির্দেশের প্রয়োজন। যতক্ষণ পর্যন্ত ক্রয়মূল্য থাকবে ততক্ষণ এক এক করে তা কমপিউটারে কী-বোর্ডের মাধ্যমে দেওয়া হবে এবং যখনই শেষ হয়ে যাবে তখন শূন্য দিয়ে বোঝানো হবে, এবারে থামা প্রয়োজন। এই ধরনের সমস্যার ক্ষেত্রে FOR-NEXT ব্যবহার করা অসুবিধাজনক। প্রোগ্রামের পর ফলাফল ছাপানো হয়েছে। এখানে সবচেয়ে প্রথম লাইনে A এবং R অক্ষর দুটি দেখানো হয়েছে কেবলমাত্র কোন মানটি কোন চলরাশির তা বোঝানোর জন্য। আসলে কিন্তু প্রোগ্রামটি চালানো হলে এই লাইনটি ছাপানো হবে না। এই লাইনটি ছাপাতে হলে তার জন্য 20 সংখ্যক লাইনের আগে একটি নির্দেশ দিতে হবে।

বেসিকের বিশেষ ধরনের কিছু নির্দেশাবলী

এর আগে বেসিক ভাষার কিছু সংখ্যক সহজ নির্দেশাবলী আলোচনা করা হয়েছে। এবারে কিছু বিশেষ ধরনের নির্দেশ নিয়ে আলোচনা করা হবে। অনেক সময়ে কোন সমস্যার সমাধান অল্প সংখ্যক নির্দেশের সাহায্যে করার জন্য কিংবা ফলাফল সুন্দরভাবে ছাপানোর জন্য এইসব নির্দেশের ব্যবহার হয়। আবার কোন বিশেষ ধরনের সমস্যা সমাধানের জন্যও এই সব নির্দেশের প্রয়োজন হয়।

DIM-নির্দেশঃ

DIM শব্দটি DIMENSION-এর প্রথম তিনটি অক্ষর। কিন্তু DIM-নির্দেশের প্রয়োজন কি ?

এমন অনেক সমস্যা আছে যেখানে একই ধরনের বিভিন্ন তথ্য কমপিউটারের স্মৃতিতে রাখার প্রয়োজন। মনে করা যাক কোনো একটি জিনিস 12 মাসের প্রতি মাসে কত বিক্রয় হয়েছে তা স্মৃতিতে রাখতে হবে। এখানে 12টি মূল্য পাওয়া যাবে এবং এগুলি একই ধরনের জিনিস অর্থাৎ সবগুলিই মূল্য, যদিও একটি মূল্য অপর একটির থেকে আলাদা হওয়াই স্বাভাবিক। কাজেই এর জন্য স্মৃতিতে 12টি ভিন্ন জায়গার দরকার। প্রত্যেকটি জায়গার জন্য প্রোগ্রামে আলাদা চলরাশির নাম ব্যবহার করলে ভিন্ন ভিন্ন 12টি চলরাশির নামের প্রয়োজন। এইভাবে এইরকম অন্য কোনো সমস্যায় একই ধরনের 1000টি জিনিস থাকলে 1000টি চলরাশির নামের দরকার হবে। কিন্তু স্মৃতিতে একই ধরনের জিনিস ভিন্ন ভিন্ন জায়গাতে রাখার জন্য প্রোগ্রামে কেবল একটি চলরাশির নামও ব্যবহার করা চলে। তখন নামের পর বন্ধনীর মধ্যে সংখ্যার উল্লেখ

করতে হবে। এবং তার সাহায্যে প্রত্যেকটিকে আলাদা ভাবে চিহ্নিত করা সম্ভব হবে। উপরের উদাহরণে বিক্রয় মূল্যের জন্য চলরাশির নাম SALE AMOUNT-এর বদলে সংক্ষেপে SALE.AMT ব্যবহার করা যেতে পারে। এরপর SALE.AMT (1) লিখে প্রথম মাসের বিক্রয় মূল্য, SALE.AMT (2)-এর সাহায্যে দ্বিতীয় মাসের বিক্রয় মূল্য এবং এইভাবে SALE.AMT (12) লিখে দ্বাদশ মাসের বিক্রয় মূল্য বোঝানো যাবে। এই পদ্ধতিকে বিন্যাস বা অ্যারে (Array) বলা হয়ে থাকে। কমপিউটারের ভাষায় SALE.AMT- কে অ্যারে চলরাশি বলা হয়। আগেই বলা হয়েছে যে একটি চলরাশির জন্য একটি জায়গাই স্মৃতিতে বরাদ্দ থাকে। কিন্তু অ্যারে চলরাশির ক্ষেত্রে তা করা হলে খুব মুশকিল। উপরের উদাহরণে SALE.AMT-এর জন্য যদি সেই একটি জায়গাই বরাদ্দ, করা হয় তাহলে 12টি আলাদা বিক্রয় মূল্য সেখানে রাখা যাবে না। কারণ একই জায়গাতে 12টি আলাদা বিক্রয় মূল্য কখনোই রাখা সম্ভব নয়। কাজেই কমপিউটারের স্মৃতিতে কতটা জায়গা রাখলে প্রত্যেকটিকে আলাদা ভাবে পাওয়া সম্ভব তা বোঝাতেই DIM-এর প্রয়োজন। উপরের উদাহরণে প্রোগ্রামে SALE.AMT ব্যবহার করার আগে নীচের নির্দেশটি দিতে হবে।

10 DIM SALE.AMT (12)

এই নির্দেশের সাহায্যে বেসিক ভাষার কমপাইলার বা ইন্টারপ্রিটারকে বলা হচ্ছে, SALE.AMT নামের অ্যারে চলরাশির জন্য 13টি জায়গা রাখতে হবে। 13টি কেন ? বেসিকে DIM-নির্দেশে অ্যারে চল-রাশির নামের পর যে সংখ্যাটি লেখা হয় সাধারণত শূন্য থেকে আরম্ভ করে সেই সংখ্যা পর্যন্ত জায়গা রাখা হয়। এখানে 12 থাকায় শূন্য থেকে শুরু করে 12 পর্যন্ত হবে। অর্থাৎ প্রথম জায়গাটি হবে SALE.AMT(0), দ্বিতীয়টি SALE.AMT (1)। এইভাবে ত্রয়োদশ জায়গাটিকে পেতে হলে SALE.AMT (12) বলে লিখতে হবে। এই উদাহরণে 12টি জায়গার প্রয়োজন। সেক্ষেত্রে DIM নির্দেশে SALE.AMT (11) লিখলেই 12টি জায়গা পাওয়া যেত। তবে সেক্ষেত্রে প্রোগ্রামে প্রথম মাসের বিক্রয় মূল্য বোঝাতে SALE.AMT (0) এবং দ্বাদশ মাসের বিক্রয় মূল্য জানার প্রয়োজন হলে SALE.AMT (11) লিখতে হবে। এতে অনেক সময়ে ভুল হওয়ার সম্ভাবনা থাকে। কাজেই প্রথম মাসকে যাতে SALE.AMT (1) হিসেবেই লেখা সম্ভব হয় সেই কারণে প্রথম জায়গাটি ফাঁকা রেখে দিলে কমপিউটারের স্মৃতি কোষের তেমন কিছু অপচয় হবে না। নীচের উদাহরণে এবারে দেখানো হচ্ছে কি ভাবে খুব সহজেই এই 12টি বিক্রয় মূল্য যোগ করে যোগফল বের করা যায়।

উদাহরণ 1.

```
10 DIM SALE.AMT (12)
20 TOTAL = 0
30 FOR I = 1 TO 12
40 INPUT SALE.AMT (I)
50 TOTAL = TOTAL + SALE.AMT (I)
60 NEXT I
70 PRINT "TOTAL SALE AMOUNT ="; TOTAL
80 END
```

এই উদাহরণে বিভিন্ন মূল্যের জন্য একটি নাম SALE.AMT ব্যবহার করা হয়েছে। এখানে SALE.AMT (0) ব্যবহার করা হয় নি। 12 মাস বোঝাতে 1 থেকে 12 নেওয়া হয়েছে। প্রত্যেকটি মূল্যকে চিহ্নিত করার দরকারে নামের পর বন্ধনীর মধ্যে যে মূল্যটি প্রয়োজন সেই সংখ্যাটি উল্লেখ করতে হবে। অর্থাৎ দ্বিতীয় মূল্যের জন্য SALE.AMT (2) লিখতে হবে এবং এইভাবে পঞ্চমের জন্য SALE.AMT (5) লেখা দরকার। কিন্তু একটি নাম ব্যবহার না করে ভিন্ন ভিন্ন নাম ব্যবহার করলে প্রোগ্রামটি কিরকম হবে এবারে তা নীচে দেওয়া হল।

উদাহরণ 2.

```
10 INPUT A1, A2 A3, A4, A5, A6
20 INPUT A7, A8, A9, A10, A11, A12
30 TOTAL = A1 + A2 + A3 + A4 + A5 + A6 + A7
40 TOTAL = TOTAL+ A8 + A9 + A10 + A11 + A12
50 PRINT "TOTAL SALE AMOUNT ="; TOTAL
60 END
```

12টির স্থানে 500টি হলে উদাহরণ 1-এ কেবলমাত্র 10 এবং 30 সংখ্যক লাইনের নির্দেশে 12-এর জায়গাতে 500 লিখতে হবে। কিন্তু উদাহরণ 2-এর ক্ষেত্রে কি হবে? 500টি সংখ্যার জন্য INPUT নির্দেশে আলাদা ভাবে 500টি নামের উল্লেখ করতে হবে। আবার ওই নামগুলি যোগ করার সময়ও তা লেখার দরকার। 500 না হয়ে 5000 হলে উদাহরণ 2-এর মত করলে তা একেবারেই অবাস্তব মনে হওয়া সম্ভব। কিন্তু যত বেশি সংখ্যাই হোক না কেন উদাহরণ-1-এর ক্ষেত্রে খুব একটা পরিবর্তনের প্রয়োজন নেই। সেখানে কেবলমাত্র 10 এবং 30 সংখ্যক লাইনের নির্দেশ পরিবর্তন করার দরকার।

এবারে DIM নির্দেশ সম্বন্ধে কিছু বলা প্রয়োজন।

1. এই নির্দেশ লেখার নিয়ম হল

লাইন সংখ্যা DIM চলরাশির নাম (সংখ্যা),

লাইন সংখ্যা এবং DIM শব্দের পর এক বা একাধিক ফাঁকা জায়গা, এরপর যে চলরাশির জন্য এই DIM প্রয়োজন তার নাম এবং সবশেষে বন্ধনীর মধ্যে যত সংখ্যক নাম প্রোগ্রামে ব্যবহৃত হবে সেই সংখ্যার উল্লেখ।

2. একটি DIM নির্দেশে একাধিক চলরাশির নাম থাকা সম্ভব। এই চলরাশি বিভিন্ন ধরনের হতে পারে।

10 DIM A(100), B(20), C$(50)

এখানে একই DIM নির্দেশে তিনটি চলরাশির নামের উল্লেখ আছে। এর মধ্যে আবার দুটি সংখ্যা চলরাশির নাম এবং একটি সারি চলরাশির নাম।

3. DIM নির্দেশে চলরাশির নামের পর যে সংখ্যার উল্লেখ থাকে প্রোগ্রামে কোনো নির্দেশেই সেই সংখ্যার থেকে বেশি বা শূন্যের থেকে কম লিখলেও হবে না। কোনো উদাহরণে DIM নির্দেশে A(100) থাকলে সেই প্রোগ্রামে কোনো নির্দেশে A(101) বা A(–1) লিখলে ভুল হবে।

4. অনেক সময়ে একটি টেবিলও একটি মাত্র নামের সাহায্যে স্মৃতিতে রাখা সম্ভব। মনে করা যাক, একটি প্রতিষ্ঠানে 10 জন বিক্রয়কারী আছে। কমপিউটারের স্মৃতিতে প্রত্যেক বিক্রয়কারীর 12 মাসের প্রত্যেক মাসের মোট বিক্রয় সঞ্চয় করা প্রয়োজন হলে এটি একটিমাত্র নাম ব্যবহার করে করা যেতে পারে। ধরা যাক, নাম দেওয়া হল SALE। এরপর বন্ধনীর মধ্যে লেখা যেতে পারে SALE(10, 12)। প্রথম সংখ্যা 10 বিক্রয়কারীর সংখ্যা বোঝাবে এবং বারো মাসের জন্য 12 ব্যবহার করা হয়েছে। এর জন্য নীচের DIM নির্দেশ দিতে হবে।

10 DIM SALE (10, 12)

প্রথম বিক্রয়কারীর পঞ্চম মাসের বিক্রয় জানতে হলে প্রোগ্রামে SALE(1, 5) নামে উল্লেখ করতে হবে। এখানে অবশ্যই ধরে নেওয়া হচ্ছে, স্মৃতিতে এই টেবিলে 1 থেকে 10 সারির প্রত্যেকটি সারিতে 1 থেকে 12 পর্যন্ত বিক্রয় মূল্য রাখা হয়েছে। 1 সংখ্যক সারিতে প্রথম বিক্রয়কারীর বারো মাসের তথ্যের জন্য 12টি জায়গা বরাদ্দ রয়েছে। এইভাবে প্রত্যেকটি সারিতেই 12টি জায়গা রাখা আছে। এবারে নীচের প্রোগ্রামে দেখানো হচ্ছে, কি ভাবে ওই টেবিলের সমস্ত তথ্য স্মৃতিতে সঞ্চয় করে প্রত্যেক বিক্রয়কারীর মোট বিক্রয়ের পরিমাণ বের করা সম্ভব।

উদাহরণ 3.

```
 10  DIM SALE (10, 12)
 20  FOR I = 1 TO 10
 30  REM I INDICATES SALESMAN NUMBER
 40  SUM = 0
 50  FOR J = 1 TO 12
 60  REM J INDICATES MONTH NUMBER
 70  INPUT SALE (I, J)
 80  SUM = SUM + SALE (I, J)
 90  NEXT J
100  PRINT "SALESMAN NO = "; I, "TOTAL SALE
     AMOUNT = "; SUM
110  NEXT I
120  END
```

5. DIM নির্দেশ প্রোগ্রামে যে কোনো জায়গাতেই থাকা সম্ভব। তবে যেসব চলরাশির জন্য DIM ব্যবহার করা হয়েছে প্রোগ্রামে সেইসব চলরাশির নামের ব্যবহারের আগের কোনো একটি নির্দেশ DIM হবে। অনেক সময়েই অনেকেই প্রোগ্রামের গোড়ার দিকেই এই নির্দেশ ব্যবহার করে থাকেন। বেসিকে 10-এর কম হলে DIM ব্যবহার না করলেও ভুল হবে না। অর্থ্যাৎ A একটি অ্যারে চলরাশি হলে এবং প্রোগ্রামে মোট 8টি বিভিন্ন A-এর প্রয়োজনে DIM A (8) লেখার দরকার হয় না।

OPTION BASE – নির্দেশঃ

DIM নির্দেশ আলোচনা করার সময়েই বলা হয়েছে যে, DIM নির্দেশে চলরাশির নামের পরে বন্ধনীর মধ্যে যে সংখ্যার উল্লেখ দেখা যায়, শূন্য থেকে আরম্ভ করে ওই সংখ্যা পর্যন্ত যত সংখ্যক জায়গা তা চলরাশিটির জন্য স্মৃতিতে রাখা থাকে। এখন প্রোগ্রামে সাধারণত 1 থেকে শুরু করে ওই সংখ্যা পর্যন্ত ব্যবহার করা হয়। কাজেই একেবারে প্রথম জায়গাটি অব্যবহৃতই থেকে যায়। এই অপচয় বন্ধ করার জন্য OPTION BASE নির্দেশের ব্যবহার, যে নির্দেশ প্রোগ্রামের প্রথম নির্দেশটি হতে পারে। এই নির্দেশ দেওয়ার নিয়ম নীচে দেখানো হলঃ

```
10 OPTION BASE 1
```

এরপর এই প্রোগ্রামে DIM নির্দেশে যত সংখ্যক চলরাশির নামেরই ব্যবহারে হোক না কেন প্রত্যেকটির জন্যই প্রোগ্রামে 1 থেকে আরম্ভ করে যে সংখ্যার উল্লেখ থাকবে সেই সংখ্যা পর্যন্ত ব্যবহার করা যাবে এবং স্মৃতিতেও আর শূন্যের জন্য কোনো জায়গা থাকবে না।

লাইব্রেরি ফাংশন

বেসিক ভাষায় নানা ধরনের গাণিতিক সমস্যা সমাধান সম্ভব। এইসব সমস্যায় বিভিন্ন ত্রিকোণমিতিক ফাংশন (যেমন, SINE, COSINE ইত্যাদি) কিংবা অন্যান্য কোনো ফাংশনও আসতে পারে। এখন SINE ফাংশনের কথাই ধরা যাক। এই নামে যে একটি ত্রিকোণমিতিক ফাংশন আছে তা বেসিক ভাষার কমপাইলার বা ইন্টারপ্রিটার কি করে বুঝবে এবং X-এর কোনো একটি মানের জন্য SINE(X)-এর মান কত হবে তাই বা কি করে জানবে?

আমরা জানি এইসব প্রত্যেকটি ত্রিকোণমিতিক ফাংশনের জন্য ভিন্ন ভিন্ন সিরিজ আছে। এই সিরিজ থেকে X-র মান জানা থাকলে ফাংশনের মান বের করা সম্ভব। কিন্তু এই সিরিজ থেকে ফাংশনের মান বের করার জন্য কিছু সংখ্যক নির্দেশের প্রয়োজন। কাজেই কোনো ফাংশনের মান বের করার দরকার হলে ওই সিরিজের জন্য যেসব নির্দেশ প্রয়োজন তা দেওয়া হবে। এখন যেসব ত্রিকোণমিতিক ফাংশন বা অন্যান্য কোনো ফাংশন খুব বেশি ব্যবহৃত হয় সেইসব ফাংশনের জন্য যে যে নির্দেশের প্রয়োজন সেইসব নির্দেশ দিয়ে তৈরি প্রোগ্রামগুলি বেসিক ভাষার কমপাইলার বা ইন্টারপ্রিটার যাঁরা সরবরাহ করেন সাধারণত তাঁরাই সরবরাহ করে থাকেন। সেক্ষেত্রে কোনো একটি প্রোগ্রামে এইসব ফাংশনের দরকার হলে আর নতুন করে ওই ফাংশনের জন্য প্রয়োজনীয় নির্দেশ দিতে হয় না। কেবলমাত্র যে নামে ওই ফাংশন সরবরাহ করা হয়েছে সেই নাম প্রোগ্রামে ব্যবহার করতে হবে। প্রোগ্রামে যে নির্দেশে ওই নাম থাকবে প্রোগ্রাম চলাকালীন সেই নির্দেশ পালন করার সময়ে ওই ফাংশনের জন্য যে-সব নির্দেশ দেওয়া আছে তা পালন করে ফাংশনের মান পাওয়া যাবে। এইসব ফাংশনকে লাইব্রেরি ফাংশন বলার কারণ কোনো একটি বিশেষ তথ্য জানার প্রয়োজনে যেমন লাইব্রেরিতে রাখা কোনো বই থেকে তা জানা সম্ভব তেমনি এইসব ফাংশনের মান বের করার দরকার হলে সংশ্লিষ্ট লাইব্রেরি ফাংশন ব্যবহার করে তা করা যায়। এবারে এরকম বহুল ব্যবহৃত কিছু লাইব্রেরি ফাংশন সম্বন্ধে আলোচনা করা হবে।

SQR(X) ঃ

SQUARE ROOT থেকে SQR শব্দটি এসেছে। নীচের উদাহরণে 1 থেকে আরম্ভ করে 100 পর্যন্ত সংখ্যার বর্গমূল বের করা হয়েছে।

উদাহরণ 1.

```
10 FOR X = 1 TO 100
20 Y = SQR(X)
30 PRINT X, Y
40 NEXT X
50 END
```

এখানে কোনো সূত্র ব্যবহার না করেই কেবলমাত্র SQR নামের পর বন্ধনীর মধ্যে সংখ্যার উল্লেখের সাহায্যেই ওই সংখ্যার বর্গমূল বের করা সম্ভব। বর্গমূল বের করার সূত্রটির জন্য যেসব নির্দেশ প্রয়োজন SQR(X) বলে ওই লাইব্রেরি ফাংশনকে ডাকলেই সেইসব নির্দেশ পালন করে তবেই ফাংশনের মান পাওয়া যাবে। এই লাইব্রেরি ফাংশনটি না থাকলে ওইসব নির্দেশগুলি এই প্রোগ্রামেই লিখতে হোত। এক্ষেত্রে X-এর মান ঋনাত্মক হওয়া সম্ভব নয়।

ABS(X) ঃ

ABSOLUTE শব্দ থেকে ABS নেওয়া হয়েছে। এখানে X সংখ্যাটি ধনাত্মক বা ঋণাত্মক হতে পারে। কিন্তু ABS(X) কেবলমাত্র সংখ্যাটির মান দেবে অর্থাৎ সংখ্যাটি ঋণাত্মক হলেও ঋণাত্মক চিহ্ন বাদ দিয়ে মান নেওয়া হবে। নীচের উদাহরণটি লক্ষ্য করা যাক।

উদাহরণ 2.

```
10 A = –2.4
20 B = 2.4
30 PRINT ABS(A), ABS(B)
40 END
```

এখানে A ঋণাত্মক হলেও ABS(A) লেখাতে 2.4 ছাপানো হবে, –2.4 নয়। B ধনাত্মক। কাজেই এক্ষেত্রেও কমপিউটার সেই একই 2.4 ছাপাবে। অর্থাৎ X ধনাত্মক বা ঋণাত্মক যাই হোক না কেন, ABS(X) সব ক্ষেত্রেই ধনাত্মক।

INT(X) ঃ

INTEGER থেকে INT শব্দ এসেছে। এই ফাংশনের কাজ হল X পূর্ণসংখ্যা বা ভগ্নাংশ যাই হোক না কেন INT(X) সব সময়েই পূর্ণসংখ্যা হবে। এই পূর্ণসংখ্যা কখনোই X-এর চেয়ে বড় হবে না। যেমন, X = 3.4 হলে, INT(X) হবে 3। এখানে 3 একটি পূর্ণসংখ্যা, এবং তা X-এর চেয়ে ছোট। কিন্তু INT(3.4) কি 2 হতে পারে? না। যদিও 2 একটি পূর্ণসংখ্যা এবং 2 সংখ্যাটিও 3.4-এর চেয়ে ছোট। মনে রাখতে হবে, সব চেয়ে বড় যে পূর্ণ সংখ্যাটি X-এর মধ্যে পাওয়া যায় INT ফাংশন তাই-ই দেবে। এই সংখ্যাটি X এর চেয়ে কখনোই বড় নয়। তা ছোট কিংবা সমান হতে পারে। X-এর মান −3.4 হলে, INT(X) কি হবে? −3, না −4? −3 হবে না, কারণ −3, X সংখ্যাটির চেয়ে বড়। কিন্তু −4, X এর চেয়ে ছোট। কাজেই INT(−3.4) হবে −4।

উদাহরণ 3.

```
10 PRINT INT(3.7) INT(6), INT(-3.7)
```

এখানে প্রথম INT ফাংশনের জন্য কমপিউটার 3 ছাপাবে, দ্বিতীয়টির জন্য 6 ছাপানো হবে। কিন্তু তৃতীয় INT ফাংশনের জন্য −4 ছাপাবে।

SGN(X) ঃ

কোনো সংখ্যা ধনাত্মক, শূন্য না ঋণাত্মক তা বোঝানোর জন্য এই ফাংশনের ব্যবহার। X ধনাত্মক হলে SGN(X), +1 দেবে, শূন্য হলে 0 এবং ঋণাত্মক হলে −1 দেবে।

উদাহরণ 4.

```
10 PRINT SGN(4), SGN(3 - 3), SGN(-4.2)
```

এই নির্দেশের জন্য ভিডিইউর পর্দায় নীচের লাইনটি ফুটে উঠবে।

```
        1    0    -1
```

SIN(X), COS(X), TAN(X) ঃ

ত্রিকোণমিতিক ফাংশন SINE, COSINE, TANGENT-এর জায়গাতে SIN, COS, TAN লেখা হয়। এখানে X রেডিয়ান হিসেবে দেওয়া থাকে। ARCTAN অর্থাৎ $TAN^{-1}(X)$-এর জন্য ATN(X) ব্যবহার করা হয়।

LOG(X), EXP(X)ঃ

Logarithm এবং exponential ফাংশন দুটি যথাক্রমে LOG এবং EXP নামে বেসিকে ব্যবহার করা হয়। LOG-এর ক্ষেত্রে ভিত্তি হল e। অর্থাৎ LOG(10) প্রোগ্রামে ব্যবহার করলে $\log_e 10$-এর মান পাওয়া যাবে। আবার EXP(2) লিখলে e^2 এর মান পাওয়া যায়।

এইসব ফাংশন যদি লাইব্রেরি ফাংশন হিসেবে না পাওয়া যেত তা হলে এদের ব্যবহার করা হোত কি ভাবে ? সেক্ষেত্রে এদের জন্য যে সব নির্দেশের প্রয়োজন তা প্রোগ্রামে লিখতে হবে। মনে করা যাক EXP(X) লাইব্রেরি ফাংশন হিসেবে দেওয়া নেই। তাহলে X-এর কোনো একটি মানের জন্য EXP(X) কি ভাবে পাওয়া যাবে ? EXP(X) অর্থাৎ e^x। এর জন্য নীচের সিরিজের সাহায্য নেওয়া যেতে পারে।

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots + \frac{x^n}{n!} + \ldots.$$

এখানে $n! = 1 \times 2 \times 3 \times \ldots \times (n-1) \times n$ বোঝানো হয়। এই সিরিজের পদ সংখ্যা অসীম। কাজেই এক্ষেত্রে একথা জানা দরকার, e^x-এর মান দশমিকের পর কত ঘর পর্যন্ত সঠিক পাওয়া প্রয়োজন। যদি দশমিকের পর চতুর্থ অঙ্ক পর্যন্ত সঠিক দরকার হয়, তাহলে যে প্রথম পদের মান .0001-এর চেয়ে ছোট হবে সেখানে থামলেই চলবে। এর কারণ এই সিরিজের বৈশিষ্ট্য, সাধারণত কোনো পদের মান .0001-এর থেকে ছোট হলে পরের পদের মান আরও ছোট হবে। কাজেই এই পদের মান যোগ করলেও e^x-এর মানের যা তফাত হবে তা দশমিকের পর পঞ্চম অঙ্কে কিছুটা হেরফের হওয়ার সম্ভাবনা থাকবে। কিন্তু দশমিকের পর চতুর্থ অঙ্ক পর্যন্ত সঠিক মান পাওয়ার দরকার বলে এই পদের এবং এর পরবর্তী পদের মান বের করার আর প্রয়োজন হবে না। এখন এই সিরিজে পরপর পদগুলি বের করার একটি সহজ উপায় নীচে দেওয়া হলঃ

প্রথম পদ = T1 = 1

দ্বিতীয় পদ = T2 = T1 . X/1

তৃতীয় পদ = T3 $= \frac{X^2}{2!} = x \cdot \frac{x}{2}$ = T2 . X/2

চতুর্থ পদ = T4 $= \frac{X^3}{3!} = \frac{x^2}{2!} \cdot \frac{x}{3}$ = T3 . X/3

পঞ্চম পদ = T4 $= \frac{X^4}{4!} = \frac{x^3}{3!} . \frac{x}{4} = T4 . X/4$

এইভাবে প্রথম পদ থেকে দ্বিতীয় পদ, দ্বিতীয় পদ থেকে তৃতীয় পদ, এইভাবে পরপর পদগুলি বের করা সম্ভব। কমপিউটারে প্রোগ্রাম লেখার সময়ে পদগুলির জন্য আর আলাদা করে T1, T2, T3, . . . নাম দেওয়ার প্রয়োজন নেই। প্রথম পদকে TERM বলে চিহ্নিত করে দ্বিতীয় পদ থেকে আরম্ভ করে অন্য পদগুলিকে নীচের সূত্র অনুসারে লেখা যেতে পারে।

TERM = TERM * X/A

প্রথমে TERM-এ এবং A তে 1 রাখা হল। কাজেই এই সূত্র অনুসারে প্রথমবার TERM-এর মান হবে 1 . X/1 অর্থাৎ X। এরপর A-র মান 1 করে বাড়িয়ে আবার এই নির্দেশে ফিরে এলে এবারে TERM-এর মান হবে X.X/2 অর্থাৎ $x^2/2!$। এইভাবে পরপর পদগুলি বের করা যাবে। X-এর একটি মান পড়ে নিয়ে e^x-এর মান কত হবে তা বেসিক ভাষায় লেখা প্রোগ্রামের সাহায্যে এবারে দেখানো যাক।

উদাহরণ 5.

```
10 INPUT X
20 TERM = 1
30 SUM =1
40 A = 1
50 TERM = TERM * X/A
60 IF ABS(TERM) <.0001 THEN 100
70 SUM = SUM + TERM
80 A = A + 1
90 GO TO 50
100 PRINT X ; "EXPONETIAL OF X = "; SUM
110 END
```

উপরের উদাহরণটি বিশ্লেষণ করে কী পাওয়া যাবে ? প্রথমে 10 সংখ্যক লাইনের নির্দেশ অনুসারে x নামের চলরাশির একটি মান পড়া হচ্ছে। মনে করা যাক তা হল .1। এরপর তিনটি নির্দেশের সাহায্যে TERM, SUM এবং A-তিনটি চলরাশিতেই 1 সংখ্যাটি রাখা হবে। 50 সংখ্যক লাইনের নির্দেশ অনুসারে TERM এবং A-তে 1 থাকায় TERM = x অর্থাৎ .1 পাওয়া যাবে। এই মান

.0001-এর থেকে ছোট কিনা পরীক্ষা করে দেখা হচ্ছে 60 সংখ্যক নির্দেশে। এবারে ছোট না হওয়ায় SUM চল-রাশিতে প্রথমে যে 1 ছিল তার সঙ্গে ওই .1 যোগ করে SUM-এ পাওয়া গেল 1.1। এরপর A-তে 1 যোগ করে A-এর মান হবে 2 এবং আবার 50 সংখ্যক লাইনে এসে TERM-এর-নতুন মান বের করা হবে। এবারে TERM-এর আগের মান .1 থাকায়, তার সঙ্গে .1 গুণ করে গুণফলকে 2 দিয়ে ভাগ করে TERM-এ পাওয়া যাবে .1×.1/2 অর্থাৎ .005। এই মান এবারও .0001-এর থেকে ছোট নয়। কাজেই আবার আগের মতই SUM-এর পুরনো মানের সঙ্গে TERM-এর নতুন মান যোগ করে হবে 1.105। এরপর A-এর মানের সঙ্গে 1 যোগ করে A তে পাওয়া যাবে 3। আবার 50 সংখ্যক লাইনে এসে কমপিউটার সেখানকার নির্দেশ পালন করে TERM-এর নতুন মান হবে .005×.1/3 অর্থাৎ .000167। এই মানও .0001-এর থেকে ছোট না হওয়ায় SUM-এর এবং A-এর নতুন মান পাওয়া যাবে যথাক্রমে 1.105167 এবং 4। এবারে 50 সংখ্যক লাইনে এসে TERM-এর নতুন মান হবে .00000417। এবারে কিন্তু এই মান .0001-এর থেকে ছোট। কাজেই এই ক্ষেত্রে আর 70, 80 এবং 90 সংখ্যক লাইনের নির্দেশ পালন না করে একেবারে 100 সংখ্যক লাইনে এসে e^x-এর মান যা SUM চলরাশিতে সঞ্চয় করা আছে তা ছাপিয়ে থেমে যাবে। কাজেই দেখা যাচ্ছে যতক্ষণ পর্যন্ত না TERM-এর মান .0001-এর থেকে ছোট হবে ততক্ষণ নতুন নতুন TERM-এর মান বের করে তা SUM-এ যোগ হবে। এখানে লক্ষ্য করার বিষয় TERM-এর মান ক্রমশই ছোট থেকে আরও ছোট হচ্ছে। কাজেই বলা যায়, যে পদের মান বের করে থামা হয়েছে তার পরের কোনো পদের মানই আর এর থেকে বড় হবে না এবং সেক্ষেত্রে ফলের দিক দিয়ে কোনো হেরফের হবে না। অর্থাৎ দশমিকের পর চতুর্থ স্থান পর্যন্ত ফল সঠিকই থাকবে।

এখানে 60 সংখ্যক লাইনে ABS ফাংশনটি কেন ব্যবহার করা হল ? x-এর মান ধনাত্মক বা ঋণাত্মক দুরকমই হওয়া সম্ভব। মনে করা যাক x-এর মান −2 এবং 60 সংখ্যক লাইনের নির্দেশে ABS ফাংশন ব্যবহার করা হয় নি। এবারে কিন্তু প্রথম বার TERM-এর মান −2 হওয়ায় তা .0001 এই ধনাত্মক সংখ্যাটি থেকে ছোট হবে। কাজেই প্রথমবারই 100 সংখ্যক লাইনের নির্দেশে এসে e^{-2}-এর মান 1 ছাপিয়ে থেমে যাবে। কিন্তু e^{-2}-এর মান কখনোই 1 হওয়া সম্ভব নয় এইজন্যই ABS ফাংশনটির ব্যবহার। TERM-এর মান-2 হলেও যদি ABS ব্যবহার করা হয় তাহলে ABS(−2) কিন্তু 2 হবে। 2 কিন্তু .0001-এর থেকে ছোট নয়। কাজেই আগের মতই 70, 80 এবং 90 সংখ্যক নির্দেশ পালন করে আবার 50 সংখ্যক

লাইনের নির্দেশে চলে আসবে। এইভাবে চলতে থাকবে যতক্ষণ পর্যন্ত না ABS(TERM)-এর মান .0001-এর থেকে ছোট হয়।

ফাংশনের মান সাধারণত কিভাবে বের করা হয় উপরের উদাহরণের সাহায্যে তার একটু আভাস দেওয়া হল। তবে লাইব্রেরি ফাংশন হিসেবে থাকলে কোনো প্রোগ্রামারকে কিন্তু সেই ফাংশনের জন্য এরকম নির্দেশ লিখে ফাংশনের মান বের করার প্রয়োজন হয় না। কেবলমাত্র ফাংশনের নামের পর বন্ধনীর মধ্যে কোনো চলরাশির নাম, কোনো সংখ্যা বা কোনো বীজগাণিতিক প্রকাশ লিখলেই তা কাজ করবে। উদাহরণ হিসেবে EXP(Y), EXP(.5) বা EXP(X + .1)-এর উল্লেখ করা যেতে পারে। প্রথমটিতে Y-এর মানের উপর নির্ভর কররে EXP(Y)-এর মান, EXP(.5)-এর বেলায় .5-এর জন্য EXP(.5)-এর মান বের করা হবে। তৃতীয়ের ক্ষেত্রে X চলরাশির বর্তমান মানের সঙ্গে .1 যোগ করার পর যে সংখ্যা পাওয়া যাবে সেই সংখ্যার জন্যই ফাংশনের মান বেরোবে।

RND(X)ঃ

এই ফাংশনটি সবচেয়ে বেশি ব্যবহৃত হয়। এই ফাংশন ব্যবহার করলে 0 এবং 1-এর মধ্যে কোনো একটি সংখ্যা পাওয়া যাবে। সংখ্যাটি কত হবে তা আগে বলা সম্ভব নয়। এটি 0 এবং 1-এর মধ্যে যে কোনো একটি সংখ্যা হতে পারে। কিন্তু কখনোই 1 হবে না। RANDOM শব্দ থেকে RND নেওয়া হয়েছে। এই ফাংশন লেখার নিয়ম হল ঃ

RND অথবা RND(X)

RND ফাংশন পরপর কয়েকবার ব্যবহার করলে কি সংখ্যা দেবে তা দেখার জন্য নীচের প্রোগ্রামের সাহায্য নেওয়া যেতে পারে।

উদাহরণ 6.

```
10 FOR I = 1 TO 10
20 PRINT RND
30 NEXT I
40 END
```

RND-এর স্থানে RND(X) লেখা সম্ভব। সেক্ষেত্রে X-এর মান ধনাত্মক, শূন্য বা ঋণাত্মক হতে পারে। X ধনাত্মক হলে RND-এর মতই ব্যবহার করবে। কিন্তু X শূন্য হলে ফাংশনটি শেষ যে সংখ্যা দিয়েছিল সেই সংখ্যাই দেবে। আবার X ঋণাত্মক হলে এই ফাংশন একটি নতুন সংখ্যা দিয়ে শুরু করে নতুন ধারাবাহিক সংখ্যা দেবে।

সারিমালা সংক্রান্ত লাইব্রেরি ফাংশন ঃ

এতক্ষণ যে সব ফাংশন সম্বন্ধে আলোচনা করা হয়েছে সে সবই সংখ্যা সংক্রান্ত। এবারে যে সব ফাংশন নিয়ে বলা হবে সেই সব ফাংশন সারিমালার উপর কার্যকর হয়।

LEN (expr $) ঃ

সারিমালায় যত সংখ্যক অক্ষর বা চিহ্ন আছে সেই সংখ্যা এই ফাংশনের সাহায্যে পাওয়া যায়।

উদাহরণ 7.

নীচের লাইনটি টিভির পর্দায় লিখলে

```
10 PRINT LEN ("HOW ARE YOU ?")
```

কমপিউটারের ভিডিইউতে 12 ফুটে উঠবে। এখানে দুটি কোটেশন চিহ্নের মধ্যে ফাঁকা জায়গা ধরে মোট 12টি অক্ষর আছে।

উদাহরণ 8.

নীচের PRINT নির্দেশের জন্য পর্দায় 5 ফুটে উঠবে। এক্ষেত্রে দুটি কোটেশনের মধ্যে মোট 5টি অক্ষর আছে।

```
10 A$ = "HAPPY"
20 PRINT LEN(A$)
30 END
```

LEFT$ (expr $, n) ঃ

এই ফাংশনের জন্য সারিমালার বাম পার্শ্ব থেকে n সংখ্যক অক্ষর বা চিহ্ন পাওয়া যাবে।

উদাহরণ 9.

```
10 A$ = "HAPPY"
20 PRINT LEFT$(A$, 2)
30 END
```

এই PRINT নির্দেশের জন্য পর্দায় HA ফুটে উঠবে। A$ এই চলরাশিতে যে সারিমালা রাখা আছে তা হল H, A, P, P এবং Y এই পাঁচটি অক্ষর। এই পাঁচটি অক্ষরের বাঁ দিকের দুটি অক্ষর হল H এবং A। কাজেই LEFT$ ফাংশনটির জন্য HAPPY শব্দের বাঁ দিকের দুটি শব্দ HA পর্দায় ফুটে উঠবে। কেন এমন হল ? LEFT$ ফাংশনে বন্ধনীর মধ্যে সারি চলরাশি বা সারিমালার উল্লেখ থাকে

এবং এরপর কমা চিহ্নের পর একটি সংখ্যার উল্লেখ করা হয়। ওই সংখ্যার সাহায্যে বোঝানো হয় সারি চলরাশিতে বা সারিমালায় যে সব অক্ষর আছে সেই অক্ষরগুলির বাঁ দিক থেকে কটি অক্ষর নেওয়া হবে। এখানে 2 থাকার A$ নামের চলরাশিতে যে সব অক্ষর আছে তার বাঁ দিক থেকে 2টি অক্ষর নেওয়া হবে।

RIGHT$ (EXPR$, n)ঃ

এটি ফাংশন LEFT$এর ঠিক বিপরীত কাজ করবে। অর্থাৎ সারিমালার ডানদিকের n সংখ্যক অক্ষর বা চিহ্ন দেবে।

উদাহরণ 10.

```
10 A$ = "HAPPY"
20 PRINT RIGHT$ (A$, 2)
30 END
```

এই প্রোগ্রাম চালানো হলে পর্দায় PY ফুটে উঠবে।

উদাহরণ 11.

```
10 A$ = "HAPPY"
20 PRINT LEFT$(A$, 3); RIGHT$(A$, 2)
30 END
```

এবারে পর্দায় কি ফুটে উঠবে ? HAPPY এর কারণ প্রথম ফাংশনটি LETT$ এবং বন্ধনীর মধ্যে 3 সংখ্যা থাকায় HAPPY শব্দের প্রথম তিনটি অক্ষর HAP লেখা হবে। এরপর RIGHT$ এবং বন্ধনীর মধ্যে 2 থাকায় PY লেখা হবে এবং দুটি ফাংশনের মধ্যে সেমিকোলন চিহ্ন থাকায় অক্ষরগুলি একেবারে ঠিক পরপর এসে যাবে।

MID$(expr$, p, n)ঃ

এই ফাংশনের কাজ হবে সারিমালার p তম অক্ষর থেকে আরম্ভ n-সংখ্যক অক্ষর দেওয়া।

উদাহরণ 12.

```
10 A$ = "HAPPY"
20 PRINT MID$(A$, 3, 2)
30 END
```

এবারে পর্দায় ফুটে উঠবে PP। HAPPY শব্দের 3 সংখ্যক অক্ষর হল P এবং এই অক্ষরটিকে ধরে 2টি অক্ষর হবে PP।

TAB (n) এবং SPC (n) ঃ

বেশ কিছু সংখ্যক লাইব্রেরি ফাংশন সম্বন্ধে ইতিপূর্বে আলোচনা করা হয়েছে। এবারে TAB এবং SPC এই লাইব্রেরি ফাংশন সম্বন্ধে বলা যাক। এই ফাংশন দুটি PRINT এবং LPRINT নির্দেশের সঙ্গে ব্যবহার করা হয়। এতক্ষণ পর্যন্ত PRINT এবং LPRINT-এ কমা এবং সেমিকোলনের ব্যবহার লক্ষ্য করা গেছে। কমা ব্যবহারে এক লাইনে 5টি ক্ষেত্র এবং প্রতি ক্ষেত্রে 14টি অক্ষর লেখার জায়গা থাকে। কিন্তু সেমিকোলন ব্যবহার করলে দুটি মানের মধ্যে কেবলমাত্র একটি ফাঁকা জায়গা পাওয়া যায়। TAB-এর সাহায্যে প্রয়োজনমত লাইনের কোনো একটি জায়গা থেকে লেখা শুরু করা সম্ভব। কিন্তু TAB ফাংশন দেওয়া হবে কী ভাবে ? TAB ফাংশন দেওয়ার নিয়ম নীচে দেওয়া হল।

TAB (n)

1. এখানে n-এর মান ভগ্নাংশে থাকলেও তা পূর্ণ সংখ্যায় পরিবর্তন করা হয়। n-এর মান যদি হয় 3.8 তাহলে তা 4 ধরা হবে। কিন্তু n-এর মান 3.3 দিলে সেক্ষেত্রে নিতে হবে 3। অর্থাৎ দশমিকের পর সংখ্যাটি 5 কিংবা তার চেয়ে বড় হলে দশমিকের আগের সংখ্যার সঙ্গে 1 যোগ হবে।

2. n- এর মান 0 বা তার চেয়ে ছোট হলে 1 ধরা হয়।

3. n, 80- এর থেকে বড় হলে n- এর মান বের করার নিয়ম হবে n = n mod 80।

অর্থাৎ n = 90 হলে n = 90 mod 80 = 10 হবে।

mod-এর ব্যবহারে সংখ্যাটিকে 80 দিয়ে ভাগ করে যে ভাগশেষ পাওয়া যাবে সেই সংখ্যাটিই n-এর মান হিসেবে নেওয়া হবে।

এবারে PRINT নির্দেশে TAB ব্যবহার করে দেখানো হবে লাইনে কি ভাবে তা ছাপানো হয়।

উদাহরণ 13.

```
10 A = 52.3
20 PRINT "1234567890123456789"
30 PRINT TAB(5); "THE VALUE OF A"
40 PRINT TAB(9); A
50 END
```

উপরের প্রোগ্রামটি চালানো হলে নীচের তিনটি লাইন পর্দায় ফুটে উঠবে।

```
1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9
        T H E   V A L U E   O F   A
                    5 2 . 3
```

এখানে প্রথম লাইনটি 20 সংখ্যক লাইনের নির্দেশের জন্য পর্দায় লেখা হবে। দ্বিতীয় PRINT নির্দেশে TAB ব্যবহার করায় THE শব্দের T অক্ষরটি দ্বিতীয় লাইনের পঞ্চম জায়গা থেকে লেখা শুরু হবে। এরপর তৃতীয় নির্দেশে TAB(9) থাকায় নবম জায়গা থেকে লেখা শুরু হওয়ার কথা। কিন্তু A একটি সংখ্যা চলরাশি এবং এর মান ধনাত্মক হওয়ায় একটি ঘর ফাঁকা থাকবে। কাজেই 52.3 লেখা দশম জায়গা থেকে শুরু হয়েছে। কেবলমাত্র কমা এবং সেমিকোলনের ব্যবহার করে PRINT নির্দেশের সাহায্যে এইরকম লাইনের বিশেষ কোনো জায়গা থেকে লেখা শুরু করা কষ্টসাধ্য ব্যাপার। 30 সংখ্যক লাইনে TAB(5)-এর জায়গায় TAB(5.4) থাকলেও সেই পঞ্চম জায়গা থেকেই লেখা শুরু হোত। কিন্তু TAB(5.8) থাকলে ষষ্ঠ জায়গা থেকে আরম্ভ হবে।

4. একটি PRINT নির্দেশে একাধিক TAB ফাংশন থাকা সম্ভব, তবে একের বেশি থাকলে পরের কোনো TAB ফাংশনে যে সংখ্যার উল্লেখ থাকবে সেই সংখ্যা আগের TAB ফাংশনের সংখ্যা থেকে অবশ্যই বড় হবে। তা না হলে পরের TAB ফাংশনের জন্য লেখা হবে পরের লাইনে।

উদাহরণ 14.

```
100 PRINT TAB(40); "A"; TAB(30); "B"
```

এক্ষেত্রে দ্বিতীয় TAB ফাংশনে সংখ্যাটি 30 থাকায় তা প্রথম TAB ফাংশনের সংখ্যাটির থেকে ছোট। কাজেই এই PRINT নির্দেশ পালন করার অর্থ, প্রথম লাইনে চল্লিশতম জায়গায় A এবং দ্বিতীয় লাইনের তিরিশতম জায়গাতে B অক্ষরটি লেখা হবে।

5. PRINT নির্দেশে একেবারে শেষে TAB ফাংশন থাকলে তারপর একটি সেমিকোলন চিহ্ন আছে ধরে নেওয়া হয়। সেক্ষেত্রে এই নির্দেশের পরে অন্য কোনো PRINT নির্দেশ ব্যবহার করলে সেখানকার চলরাশির মান লেখার সময় তা আগের লাইনেই লেখা হবে।

উদাহরণ 15

```
100 PRINT TAB(10); "A"; TAB(25)
110 PRINT "B"
```

এই PRINT নির্দেশ দুটির জন্য একটিই লাইন পর্দায় ফুটে উঠবে। দশম স্থানে A অক্ষরটি এবং পঞ্চবিংশতি স্থানে B অক্ষরটি লেখা হবে। 100 সংখ্যক লাইনের PRINT নির্দেশে একেবারে শেষে

একটি TAB ফাংশন থাকায় বেসিক কমপাইলার বা ইন্টারপ্রিটার ধরে নেবে এরপর একটি সেমিকোলন আছে। আর সেমিকোলন আছে বলে দ্বিতীয় লাইনে না লিখে B এই একই লাইনে লেখা হবে। একটি PRINT নির্দেশের শেষে সেমিকোলন থাকলে সেই লাইনেই পরের PRINT নির্দেশের চলরাশির মান লেখা হতে থাকে।

উদাহরণ 16.

```
10 A = 20
20 I = 1 TO 5
30 PRINT A;
40 A = A + 2
50 NEXT I
60 END
```

এই উদাহরণে PRINT নির্দেশে A-এর পর সেমিকোলন থাকায় প্রত্যেকটি A-এর মানই একই লাইনে হবে। লাইনে কি ভাবে তা লেখা হবে ?

2 4 6 8 10

প্রথমে A-এর মান 2 থাকায় 2 লিখবে। এরপর A-এর মান 4 হবে এবং FOR নির্দেশে I-এর মান 2 হবে এবং তা 5-এর থেকে কম হওয়াতে আবার PRINT নির্দেশ পালন করে ওই একই লাইনে সংখ্যাটি ধনাত্মক হওয়ায় একটি জায়গা ফাঁকা রেখে 4 লেখা হবে। এইভাবে একে একে A-এর পাঁচটি মানই লিখে প্রোগ্রামটি থেমে যাবে। সেমিকোলনের জায়গায় কমা চিহ্ন থাকলে 5টি মানই এক লাইনে লেখা হবে। তবে কমার জন্য এক একটি ক্ষেত্রে (একটি ক্ষেত্রে 14টি অক্ষর লেখার মত জায়গা থাকে) এক একটি সংখ্যা লেখা হবে। কাজেই PRINT নির্দেশে শেষে TAB ফাংশন থাকলে এরপর সেমিকোলন চিহ্ন আছে ধরে নেওয়া হয় এবং সেইজন্য পরের PRINT নির্দেশের জন্যও ওই আগের লাইনেই লেখা হয়।

TAB ফাংশনের মত SPC ফাংশন লেখারও নিয়ম হল ঃ

SPC(n)

1. এক্ষেত্রেও n-এর মান ভগ্নাংশ হলে তাকে পূর্ণ সংখ্যায় পরিবর্তন করা হয়। অর্থাৎ 3.8 হলে তা 4 হবে। কিন্তু 3.4 হলে 3 ধরা হয়। n পূর্ণসংখ্যার রূপান্তর করার পর এই ফাংশনের কাজ হল n সংখ্যক জায়গা ফাঁকা রাখা।

উদাহরণ 17.

```
10 PRINT "1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9"
20 PRINT TAB(4) ; "A"; SPC(5) ; "B"
```

```
30 END
```

এই প্রোগ্রামের জন্য নীচের লাইন দুটি লেখা হবে

```
1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9
    A          B
```

SPC ফাংশনে n-এর মান 5 থাকায় A লেখার পর 5টি জায়গা ফাঁকা রেখে B লেখা হয়েছে।

2. n-এর মান শূন্য বা ঋণাত্মক হলে SPC ফাংশনের কাজ হবে না অর্থাৎ কোনো জায়গা ফাঁকা রাখা হবে না।

3. n, 80-এর চেয়ে বড় হলে TAB ফাংশনের মতই n-এর মান হবে n mod 80।

4. একটি PRINT নির্দেশে একাধিক SPC ফাংশন থাকা সম্ভব। SPC ফাংশনে যে সংখ্যার উল্লেখ থাকে তার জন্য যদি পর্যাপ্ত পরিমাণ জায়গা সেই লাইনে না থাকে তাহলে নিজের থেকেই পরের লাইনে গিয়ে পরের সংখ্যা বা অক্ষর ছাপাবে।

উদাহরণ 18.

```
100 PRINT "A"; SPC(35) ; "B"; SPC(50) ; "C"
```

এখানে লাইনের প্রথম স্থানে A অক্ষরটি ছাপাবে। এরপর 35টি জায়গা ফাঁকা রেখে 37-তম স্থানে B অক্ষর পর্দায় লেখা হবে। পরের SPC ফাংশনে 50 থাকায় মোট 37 + 50 = 87 হচ্ছে। কিন্তু এক লাইনে 80টি জায়গা থাকায় C অক্ষরটি ওই লাইনে না লিখে পরের লাইনে 7টি ফাঁকা জায়গা রেখে অষ্টম স্থানে লেখা হবে।

5. একই PRINT নির্দেশে TAB এবং SPC দুটি ফাংশনই লেখা সম্ভব।

উদাহরণ 19.

```
10 PRINT TAB (40); "*"
20 FOR I = 1 TO 5
30 PRINT TAB (40 – I); "*"; SPC(2*I – 1); "*"
40 NEXT I
50 PRINT TAB (34); "* * * * * * * * * * * * *"
60 END
```

```
            *
          *   *
        *       *
      *           *
    *               *
  *                   *
* * * * * * * * * * * * *
```

এই প্রোগ্রাম চালানো হলে যে ছবি পাওয়া যাবে তা নীচে দেওয়া হল।

PRINT USING নির্দেশঃ

এই নির্দেশের সাহায্যে চলরাশির মান লাইনের সঠিক জায়গাতে লেখা সম্ভব। TAB ফাংশনের সাহায্যে তা লাইনের কোন জায়গা থেকে শুরু হবে তা বলা যায়। কিন্তু সংখ্যাটি কোথায় শেষ হবে তা অনেক সময়ে বলা সম্ভব হয় না। PRINT USING নির্দেশের সাহায্যে দশমিকের পর কত ঘর পর্যন্ত রাখা হবে তাও নির্দেশে বলে দেওয়া হয়। এরফলে লাইনে কোথায় সংখ্যাটি লেখা শেষ হবে তা বলা চলে। এই নির্দেশে লেখার নিয়ম হল ঃ

PRINT USING সারি রাশিমালার প্রকাশ; এক বা একাধিক চলরাশি

উদাহরণ 1.

```
10 A = 15.46
20 B = 482.513
30 C$ = "# # # . # #   # # # . # #"
40 PRINT USING C$ ; A, B
50 END
```

এই প্রোগ্রামে চালানো হলে নীচের লাইনটি পর্দায় ফুটে উঠবে

```
15.46    482.51
```

30 সংখ্যক লাইনের নির্দেশে C$ একটি সারি চলরাশি। এই সারি চলরাশিতে সারি রাশিমালা রাখা হচ্ছে। তবে এইভাবে না লিখে PRINT USING নির্দেশেই সারি রাশিমালা প্রকাশ করা যেত। যেমন–

```
30 PRINT USING " # # # . # #   # # # . # #" ; A, B
```

এইভাবে লিখলে আর সারি চলরাশি ব্যবহারের প্রয়োজন নেই। এখানে উদ্ধৃতি চিহ্নের মধ্যে সংখ্যা-চিহ্নের ব্যবহার দিয়ে বোঝানো হচ্ছে কত সংখ্যক অঙ্ক লেখা হবে। দশমিক চিহ্নের আগে তিনটে অঙ্ক এবং পরে দুটি অঙ্ক–এরকম দুটি সংখ্যা এক লাইনে লেখা সম্ভব।

PRINT USING নির্দেশ সারি রাশিমালা এবং সংখ্যা সুন্দরভাবে ছাপানোর জন্য ব্যবহার করা হয়। এর জন্য বিশেষ বিশেষ চিহ্ন দিয়ে বোঝানো হয়ে থাকে কিভাবে পর্দায় বা প্রিন্টারে লাগানো কাগজে অক্ষর বা সংখ্যা ছাপানো হবে। প্রথমে সারি রাশিমালা লেখার জন্য যে-সব চিহ্ন ব্যবহার করা হয় তা আলোচনা করা যাক।

"!"-এই চিহ্ন দিয়ে বোঝানো হয় যে, সারি রাশিমালার কেবলমাত্র প্রথম অক্ষরটি ছাপানো হবে ।

উদাহরণ 2.

```
10 A$ = "FIRST"
20 PRINT USING "!"; A$
30 END
```

এক্ষেত্রে F অক্ষরটিই লিখবে ।

"&"-এর ফলে সমস্ত রাশিমালাই লেখা হবে ।

উদাহরণ 3.

```
10 A$ = "FIRST"
20 PRINT USING "&"; A$
30 END
```

এবারে FIRST লেখা হবে ।

\n spaces\ — "\"-এই দুটি চিহ্নের মধ্যে n সংখ্যক জায়গা ফাঁকা থাকলে মোট 2 + n সংখ্যক অক্ষর রাশিমালাটি থেকে ছাপানো হবে ।

উদাহরণ 4.

```
10 A$ = "CONTINUATION"
20 B$ = "\\"
30 C$ = "\   \"
40 D$ = "\        \"
50 PRINT USING B$; A$
60 PRINT USING C$; A$
70 PRINT USING D$; A$
80 END
```

এই প্রোগ্রাম চালানো হলে নীচের লাইনগুলি পর্দায় ফুটে উঠবে ।

```
CO
CONTI
CONTINUATI
```

কেন এমন হল ? B$-এ দুটি "\" চিহ্নের মধ্যে কোনো ফাঁকা জায়গা না থাকায় 50 সংখ্যক লাইনের নির্দেশ অনুসারে A$ সারি চলরাশির প্রথম দুটি অক্ষর অর্থাৎ CO লেখা হবে । এরপর C$-এ দুটি "\" চিহ্নের মধ্যে মোট তিনটি ফাঁকা জায়গা থাকায় 60 সংখ্যক লাইনের নির্দেশ পালন করার অর্থ হবে A$ সারি চলরাশির

3 + 2 বা 5টি অক্ষর ছাপানো। এর ফলে CONTI দ্বিতীয় লাইনে লেখা হবে। তৃতীয় PRINT USING নির্দেশে D$ আছে এবং D$-এ দুটি "\" চিহ্নের মধ্যে মোট 8টি ফাঁকা জায়গা পাওয়া যাচ্ছে। কাজেই এবারে A$ সারি চলরাশির প্রথম 10টি অক্ষর অর্থাৎ CONTINUATI লেখা হবে।

উদাহরণ 5.

```
10 A$ = "FIRST"
20 B$ = "PROBLEM"
30 PRINT USING "\  \"; A$; B$
40 PRINT USING "\      \"; A$; B$
50 PRINT USING "&"; A$;
60 PRINT USING "!"; B$
70 END
```

এক্ষেত্রে নীচের লাইনগুলি ছাপানো হবে।

```
FIRST PROBL
FIRST   PROBLEM
FIRST P
```

50 সংখ্যক এবং 60 সংখ্যক লাইনের নির্দেশ দুটির জন্য তৃতীয় লাইনটি পাওয়া গেল। 50 সংখ্যা লাইনে '&" থাকায় A$ চলরাশির মান FIRST পাওয়া যাবে। A$ এর পর সেমিকোলন থাকায় পরের PRINT USING নির্দেশের জন্যও এই লাইনেই ছাপা হবে। আবার 60 সংখ্যক লাইনের নির্দেশে '!' থাকায় B$ চলরাশির মানের প্রথম অক্ষর P ঠিক T-এর পরেই এসেছে।

এবারে সংখ্যা ছাপানোর জন্য যে-সব বিশেষ চিহ্ন ব্যবহৃত হয় তা আলোচনা করা হবে।

#(সংখ্যা চিহ্ন) — প্রত্যেকটি অঙ্কের জন্য একটি সংখ্যা চিহ্ন ব্যবহার করা হয়। যতগুলি সংখ্যা চিহ্ন দেওয়া আছে, সংখ্যাটি তার চেয়ে ছোট হলে ডান দিক থেকে সংখ্যাটি লেখা হয় এবং বামদিকে ফাঁকা জায়গা থাকে।

উদাহরণ 6.

```
10 A$ = "# # # # #"
20 PRINT USING A$ ; 452
30 END
```

এখানে সংখ্যাটি তিন অঙ্কের এবং মোট পাঁচটি সংখ্যা চিহ্ন

আছে। কাজেই বামদিকে দুটি ফাঁকা জায়গা রেখে তিন অঙ্কের সংখ্যাটি লেখা হবে।

.(দশমিক চিহ্ন)—একটি সংখ্যার যে কোনো জায়গাতেই দশমিক চিহ্ন বসতে পারে।

উদাহরণ 7.

```
10 PRINT USING "# # . # #"; .58
20 PRINT USING "# # . # #"; 98.762
30 PRINT USING "# # . # #"; 23.578
```

প্রথম PRINT USING নির্দেশের জন্য 0.58 লেখা হবে। দশমিক চিহ্নের আগে সংখ্যা চিহ্ন থাকলে দশমিকের আগে সব সময়েই একটি অঙ্ক লেখা হবেই। কাজেই সংখ্যাটি .58 হওয়ায় দশমিক চিহ্নের আগে 0 বসেছে। দ্বিতীয় নির্দেশটির জন্য 98.76 লেখা হবে। এখানে দশমিকে চিহ্নের পর কেবলমাত্র দুটি অঙ্ক লেখার জায়গা থাকায় তৃতীয় অঙ্কটি লেখা সম্ভব নয়। 30 সংখ্যক লাইনের নির্দেশের জন্য 23.58 লেখা হবে। এক্ষেত্রেও দশমিক চিহ্নের পর দুটি সংখ্যা-চিহ্ন থাকায় কেবলমাত্র দুটি অঙ্কই লেখা হয়েছে। কিন্তু এবারে তৃতীয় অঙ্কটি 5-এর বেশি হওয়ায় 7-এর সঙ্গে 1 যোগ হয়ে অঙ্কটি 8 হয়েছে। কাজেই 23.57 না লিখে সংখ্যাটি 23.58 হবে।

+(যোগ চিহ্ন)—সংখ্যা-চিহ্নের শুরুতে বা শেষে যোগ চিহ্ন লেখা হয়ে থাকে। সংখ্যাটি ধনাত্মক হলে লেখার সময়ে '+' চিহ্ন বসবে। কিন্তু ঋণাত্মক সংখ্যার ক্ষেত্রে '–' চিহ্ন লেখা হবে।

উদাহরণ 8.

```
10 PRINT USING "+ # # . # #"; –23.45, 5.6
```

এই নির্দেশের ফলে যা লেখা হবে তা হলঃ

```
–23.45       +5.6
```

–(বিয়োগ চিহ্ন)—যোগ চিহ্নের মত বিয়োগ চিহ্নও শুরুতে বা শেষে লেখা সম্ভব। অবশ্য সংখ্যাটি ধনাত্মক হলে কোনো চিহ্নই বসবে না। কিন্তু ঋণাত্মক সংখ্যার ক্ষেত্রে '–' চিহ্ন বসবে। ধনাত্মক সংখ্যা হলে চিহ্নের জায়গাতে একটি ফাঁকা জায়গা থাকবে।

উদাহরণ 9.

```
10 PRINT USING "–# # . # #"; –23.45, 5.6
```

এবারে যা লেখা হবে তা নীচে দেখানো হচ্ছে।

```
–23.45        5.6
```

**(জোড়া গুণ-চিহ্ন)—সংখ্যা ছাপানোর জন্য যে সারি রাশিমালা ব্যবহার করা হয় তার প্রথম দুটি গুণ-চিহ্ন হলে সংখ্যাটি ছাপানোর

সময়ে প্রথম দিকের ফাঁকা জায়াগায় এই চিহ্ন বসে। ফাঁকা জায়গায় যাতে কোনো সংখ্যা বসানো সম্ভব না হয় তার জন্য এর ব্যবহার।

উদাহরণ 10.

10 PRINT USING "* * # # . # "; 46.78, –2.3, 765.2

এই নির্দেশের জন্য নীচের লাইনটি পর্দায় লেখা হবে

**46.8 **–2.3 *765.2

$$(জোড়া ডলার চিহ্ন)—এই দুটি চিহ্ন ব্যবহার করে সংখ্যাটির ঠিক বামদিকে ডলার চিহ্ন পাওয়া যায়। এই চিহ্ন দিয়ে বোঝানো হয় যে সংখ্যাটি একটি অর্থের পরিমাণ প্রকাশ করছে।

উদাহরণ 11.

10 PRINT USING "$ $ # # . #"; 46.78, 1.5

এবারে যে লাইনটি লেখা হবে তা হল ঃ

$46.8 $1.5

**$—এর আগের দু ধরনের চিহ্ন যে কাজ করত দুটি গুণ-চিহ্ন এবং একটি ডলার চিহ্ন ব্যবহার করে সেই কাজ করা সম্ভব। অর্থাৎ এর সাহায্যে প্রথম দিকে কোনো ফাঁকা জায়গা থাকলে সেখানে গুণ-চিহ্ন বসাবে এবং সংখ্যাটির ঠিক আগে ডলার চিহ্ন হবে।

উদাহরণ 12.

10 PRINT USING "* * $ # # # . # #"; 13.42

এই নির্দেশের জন্য নীচের লাইনটি পর্দায় দেখা যাবে।

*** $13.42

**$—এর জন্য আরও তিনটি সংখ্যা লেখা যেতে পারে। অর্থাৎ উপরের উদাহরণে সংখ্যা-চিহ্নের ব্যবহারে বোঝানো হচ্ছে, দশমিকের আগে তিনটি সংখ্যা এবং দশমিকের পর দুটি সংখ্যা। কিন্তু সংখ্যা-চিহ্নের আগে দুটি গুণ চিহ্ন এবং একটি ডলার চিহ্ন থাকায় দশমিকের আগে তিনটির বদলে ছটি পর্যন্ত সংখ্যা হওয়া সম্ভব। ছটি সংখ্যা থাকলে সবকটি সংখ্যাই পাওয়া যাবে।

,(কমা-চিহ্ন)—দশমিক চিহ্নের বামদিক কমা-চিহ্ন থাকলে দশমিকের বামদিকে প্রতি তিনটি সংখ্যার পর কমা-চিহ্ন বসবে। কমা সংখ্যা-চিহ্নর একেবারে শেষে ব্যবহার করলে সংখ্যা লেখার পর একটি কমা-চিহ্ন লেখা হবে। ক্যারেটের (পরে এই চিহ্ন সম্বন্ধে বলা হচ্ছে) সঙ্গে কমা ব্যবহারে কোনো কাজ হয় না।

উদাহরণ 13.

10 PRINT USING "# # # # # # # , . # #"; 3456789.12

20 PRINT USING "# # # # . # # ,"; 4567.23

উপরের নির্দেশগুলির জন্য নীচের লাইন দুটি পাওয়া যাবে।

3,456,789.12

4567.23

প্রথম নির্দেশে কমা-চিহ্ন ব্যবহারে একটি বড় সংখ্যা পড়ায় সুবিধে হয়।

^ ^ ^ ^—চারটে ক্যারাট চিহ্ন – কোনো সংখ্যার সূচক বোঝানোর জন্য এর ব্যবহার। চারটে ক্যারাট চিহ্ন E + xx-এর এই চারটে জায়গার জন্য। E, 10-এর বদলে লেখা হয়। xx-এর অর্থ এখানে দুটি সংখ্যা হতে পারে।

উদাহরণ 14.

10 PRINT USING "# # . # # ^ ^ ^ ^"; 456.72, 12.32 এই নির্দেশের জন্য যে লাইনটি পাওয়া যাবে তা হল

4.57 E+02 1.23 E+01

PRINT USING নির্দেশে একটি 5 অঙ্কের এবং অন্যটি 4 অঙ্কের সংখ্যা থাকা সত্ত্বেও দুটি তিন অঙ্কে সংখ্যা ছাপানো হয়েছে। এর কারণ দুটি উদ্ধৃতি চিহ্নের মধ্যে দশমিকের আগে দুটি সংখ্যা-চিহ্ন এবং পরে দুটি সংখ্যা-চিহ্ন থাকায় সংখ্যাটি ধনাত্মক না ঋনাত্মক বোঝানোর জন্য একটি জায়গা ব্যবহার করলে মোট তিনটে জায়গা থাকে।

DEF FN ... নির্দেশঃ

DEFINE FUNCTION শব্দ দুটি থেকে DEF FN নেওয়া হয়েছে। এই নির্দেশ লেখার নিয়ম হলঃ

DEF FN name (a1, a2, ...) = সংখ্যা বা রাশিমালার প্রকাশ

name যে কোনো একটি চলরাশির নাম হওয়া সম্ভব। এই নামে ফাংশনটিকে প্রোগ্রামে ব্যবহার করা যাবে। নামের পর বন্ধনীর মধ্যে এক বা একাধিক চলরাশির নাম থাকতে পারে। এই ধরনের ফাংশন প্রোগ্রামে লাইব্রেরি ফাংশনের মত ঠিক একই ভাবে ব্যবহার করা হয়ে থাকে। তবে এই ফাংশনগুলিকে ব্যবহার করার আগে প্রোগ্রামটি যিনি লিখছেন তাঁকেই লিখে নিতে হয়। সংখ্যা এবং সারিমালা সংক্রান্ত দু রকম ফাংশন হিসেবেই এদের লেখা সম্ভব। মনে করা যাক, একটি প্রোগ্রামে $\sqrt{a^2+b^2}$, $\sqrt{c^2+d^2}$, $\sqrt{u^2+v^2}$, এবং $\sqrt{p^2+q^2}$ a, b, c, d, u, v, p এবং q-এর কোনো মানের জন্য বের করা দরকার। সেক্ষেত্রে বেসিক ভাষায় এদের লিখতে গেলে কি ভাবে লেখা হবে নীচে তা দেখানো হচ্ছে।

$\sqrt{a^2+b^2}$ = SQR(A*A+B*B)

$\sqrt{c^2+d^2}$ = SQR(C*C+D*D)

এইভাবে বারবার এতগুলি অক্ষর না লিখে সংক্ষেপে কিভাবে লেখা সম্ভব তা নীচের প্রোগ্রামে দেখানো হচ্ছে।

উদাহরণে 1.

```
10 DEF FNSQ(X, Y) = SQR(X*X + Y*Y)
20 INPUT A, B
30 X1 = FNSQ(A, B)
40 . .
50 X2 = FNSQ(C, D)
60 . .
70 X3 = FNSQ(U, V)
80 . .
90 X4 = FNSQ(P, Q)
```

এই উদাহরণে 20 সংখ্যক লাইনে A এবং B এই চলরাশির নাম পড়া হল। এরপরের লাইনের নির্দেশ অনুসারে $\sqrt{A2+B2}$ বের করার জন্য 10 সংখ্যক লাইনে যে নাম ব্যবহার করা হয়েছিল সেই নাম অর্থাৎ FNSQ ব্যবহার করেই এই কাজটি করা সম্ভব। 10 সংখ্যক লাইনে ফাংশনের নামের পর বন্ধনীর মধ্যে যত সংখ্যক চলরাশির নাম থাকবে এই ফাংশনের নাম প্রোগ্রামে অন্যত্র ব্যবহার করার সময়েও ঠিক ততসংখ্যক নামেরই প্রয়োজন। এই নামগুলির একটিকে অন্যের থেকে আলাদা করার জন্য কমা চিহ্নের ব্যবহার হয়। এখানে দুটি চলরাশির নাম (X এবং Y) থাকায় এই ফাংশনকে ডাকার সময়েও দুটি নামই ব্যবহার করা হয়েছে।

এইসব ফাংশনের নামের শুরুতে সব সময়েই FN অক্ষর দুটি লিখতে হয়। এই কারণেই কোনো চলরাশির নাম FN দিয়ে শুরু করা যায় না।

DEF নির্দেশঃ

এর আগে আলোচনা করা হয়েছে যে বিভিন্ন ধরনের চলরাশি বোঝাতে চলরাশির নামের শেষের অক্ষরটির পরে ভিন্ন ভিন্ন বিশেষ চিহ্ন ব্যবহার করা হয়। যেমন, কোনো চলরাশিতে কেবলমাত্র অখণ্ড সংখ্যা রাখার জন্য সেই চলরাশির নামের শেষে শতকরা চিহ্ন (%) ব্যবহৃত হয়। কিন্তু এইসব বিশেষ চিহ্ন ব্যবহার না করেও

ভিন্ন ভিন্ন ধরনের চলরাশি বোঝানো সম্ভব। উদাহরণ হিসেবে বলা যেতে পারে

DEFINT চলরাশির নাম

DEFINT শব্দের পর যেসব চলরাশির নাম থাকবে সেসব চলরাশিতেই অখণ্ড সংখ্যা রাখা যাবে। INT অক্ষর তিনটি INTEGER শব্দের বদলে লেখা হয়েছে।

উদাহরণ 2.

DEFINT A, B, M–Q

যে প্রোগ্রামে এই নির্দেশ থাকবে সেই প্রোগ্রামে যেসব চলরাশির নাম A, B, M, N, O, P, বা Q অক্ষর দিয়ে শুরু হয়েছে সে-সব চলরাশির জায়গাতে কেবলমাত্র অখণ্ড সংখ্যা রাখা যাবে। এখানে M–Q দিয়ে বোঝানো হচ্ছে, M থেকে শুরু করে Q পর্যন্ত সব অক্ষর। এক্ষেত্রে আর ওইসব চলরাশির নামের শেষে শতকরা চিহ্ন (%) দেওয়ার প্রয়োজন হবে না।

অখণ্ড সংখ্যা ছাড়াও অন্যান্য ধরনের চলরাশির জন্যও DEF-এর সাহায্যে এইভাবেই করা যায়। যেমন–

DEFSNG চলরাশির নাম
DEFDBL চলরাশির নাম
DEFSTR চলরাশির নাম

DEFSNG ব্যবহার করা হয় একক-দৈর্ঘ্য সংখ্যা বোঝাতে। DEFDBL এবং DEFSTR যথাক্রমে দ্বি-দৈর্ঘ্য সংখ্যা এবং সারিমালা বোঝানোর জন্য ব্যবহৃত হয়। এইভাবে DEF নির্দেশ দেওয়া হলে আর ওইসব বিভিন্ন ধরনের চলরাশি বোঝাতে কোনো বিশেষ চিহ্নের প্রয়োজন হয় না।

GOSUB, RETURN নির্দেশঃ

অনেক সময়ে একই ধরনের কিছু সংখ্যক নির্দেশ প্রোগ্রামে বিভিন্ন জায়গাতে করার প্রয়োজন হয়। উদাহরণ হিসেবে বলা যেতে পারে e^x, x-এর বিভিন্ন মানের জন্য প্রোগ্রামে ভিন্ন ভিন্ন জায়গাতে বের করার দরকার হতে পারে। এখন e^x-এই লাইব্রেরি ফাংশন যদি কমপাইলারের সঙ্গে না পাওয়া যায় তাহলে প্রোগ্রামারকে e^x বের করার জন্য যে নির্দেশ দরকার তা বারবার ব্যবহার করতে হবে। এর ফলে প্রোগ্রামটি আকারে অহেতুক বড় হয়ে যাবে। প্রোগ্রামে একই ধরনের নির্দেশ যাতে বারবার না লিখতে হয় তার জন্য GOSUB এবং RETURN নির্দেশের প্রয়োজন। এক্ষেত্রে ওই নির্দেশগুলি কেবলমাত্র প্রোগ্রামের শেষের

দিকে লেখা হয়। এই নির্দেশগুলির শেষে RETURN নির্দেশটি দেওয়া থাকে। এবারে প্রোগ্রামে যে-সব জায়গাতে e^x-এর মানের প্রয়োজন সেখানে e^x বের করার নির্দেশগুলির পরিবর্তে কেবলমাত্র GOSUB নির্দেশটি দেওয়া হয়। এর ফলে প্রোগ্রামটি আগের থেকে আকারে অনেক ছোট হয়ে যাবে। নীচের উদাহরণে GOSUB-এর ব্যবহার পরিষ্কার করে বোঝানো হচ্ছে।

উদাহরণ 1.

```
10 X = 1.5
20 GOSUB 200
30 . . .
40 . . .
50 X = -1
60 GOSUB 200
70 . . .
80 . . .
90 . . .
100 . . .
110 X = 2
120 GOSUB 200
130 . . .
140 . . .
150 X = 2.1
160 GOSUB 200
170 . . .
180 . . .
190 END
200 TERM = 1
210 SUM = 1
220 A = 1
230 TERM = TERM * X/A
240 IF ABS(TERM) <.0001 THEN 280
250 SUM = SUM + TERM
260 A = A + 1
270 GO TO 230
280 RETURN
```

e^x কিভাবে বের করা হয় তা লাইব্রেরি ফাংশনের উদাহরণ 5-এ পরিষ্কার ভাবে বোঝানো হয়েছে। এখানে ওই প্রোগ্রামে যে-সব নির্দেশ ছিল তার প্রথম এবং শেষের নির্দেশ দুটি ছাড়া আরসব নির্দেশই 200 সংখ্যক লাইনের নির্দেশ থেকে শুরু করে 270 সংখ্যক লাইনে দেওয়া আছে। ওই প্রোগ্রামে 10 সংখ্যক লাইনের নির্দেশ x-এর মান জানার জন্য ব্যবহার করা হয়েছিল। এই উদাহরণে 10 সংখ্যক লাইনের নির্দেশ অনুসারে x-এর মান 1.5। এরপর GOSUB 200 নির্দেশের জন্য x-এর মান 1.5 নিয়ে 200 সংখ্যক লাইনে গিয়ে সেখানকার নির্দেশ পালন করবে এবং এখানেও ABS(TERM)-এর মান যতক্ষণ পর্যন্ত না .0001-এর থেকে ছোট হচ্ছে ততক্ষণ e^x সিরিজের পদগুলির মান বের করবে। কিন্তু যে মুহূর্তে তা ছোট হবে 280 সংখ্যক লাইনে RETURN নির্দেশ থাকায় তখনই তা সরাসরি 30 সংখ্যক লাইনের নির্দেশে চলে আসবে এবং 30 ও 40 সংখ্যক লাইনে যা নির্দেশ থাকবে তা পালন করবে। সেখানকার নির্দেশে নিশ্চই e^x-এর এই মান নিয়ে কোনো কিছু কাজ করা হবে। এরপর আবার x-এর মান –1 ধরে 60 সংখ্যক লাইনের নির্দেশ অনুসারে 200 সংখ্যক লাইনের নির্দেশে চলে আসবে। এবারে x এর মান –1 এর জন্য e^x বের করে 70 সংখ্যক লাইনের নির্দেশে ফিরে আসবে। এইভাবে যতবার GOSUB 200 নির্দেশটি ব্যবহার করা হবে ততবারই 200 সংখ্যক লাইনে এসে 270 সংখ্যক লাইনের নির্দেশ পর্যন্ত পালন করে x-এর কোনো একটি মানের জন্য e^x বের করবে। মান বের করার পর কিন্তু প্রত্যেকবারই একই লাইন সংখ্যায় যাবে না। যে GOSUB নির্দেশ থেকে এই নির্দেশগুলি পালন করার জন্য এসেছে ঠিক তার পরের নির্দেশে ফিরে যাবে। প্রথমবার 30 সংখ্যক লাইনে, দ্বিতীয়বার 70 সংখ্যক লাইনে, তৃতীয়বার 130 সংখ্যক লাইনে এবং শেষবার 170 সংখ্যক লাইনের নির্দেশ যাবে। কিন্তু GOSUB নির্দেশ ব্যবহার না করলে কি হোত ? 200 সংখ্যক লাইন থেকে আরম্ভ করে 270 পর্যন্ত যে-সব নির্দেশ আছে সে-সব নির্দেশই বারবার অর্থাৎ এই প্রোগ্রামের জন্য মোট চারবার লিখতে হোত। সেভাবে প্রোগ্রাম লিখলে কেমন দেখাবে তা নীচের প্রোগ্রাম থেকে বোঝা যাবে।

উদাহরণ 2.

```
10 X = 1.5
20 TERM = 1
30 SUM = 1
40 A = 1
50 TERM = TERM * X/A
```

```
60 IF ABS(TERM) <.0001 THEN 100
70 SUM = SUM + TERM
80 A = A + 1
90 GO TO 50
100 ..
110 ..
120 X = -1
130 TERM = 1
140 SUM = 1
150 A = 1
160 TERM = TERM * X/A
170 IF ABS(TERM) <.0001 THEN 210
180 SUM = SUM + TERM
190 A = A + 1
200 GO TO 160
210 ..
220 ..
230 ..
240 ..
250 X = 2
260 TERM = 1
270 SUM = 1
280 A = 1
290 TERM = TERM * X/A
300 IF ABS(TERM) <.0001 THEN 340
310 SUM = SUM + TERM
320 A = A + 1
330 GO TO . 290
340 ..
350 ..
360 X = 2.1
370 TERM = 1
380 SUM = 1
```

```
390 A = 1
400 IF ABS(TERM) <.0001 GO TO 440
410 SUM = SUM + TERM
420 A = A + 1
430 GO TO 400
440 ..
450 ..
460 END
```

এখানে লক্ষ্য করার বিষয় লাইন সংখ্যা আলাদা হলেও একই ধরনের নির্দেশ 20 থেকে 90, 130 থেকে 200, 260 থেকে 330 এবং 370 থেকে 430 লাইনে দেখা যাচ্ছে। GOSUB নির্দেশ না দেওয়ার জন্যই এই নির্দেশগুলি বারবার দিতে হচ্ছে। এতে প্রোগ্রাম আকারে অনেক বড় হয় এবং কমপাইলার বেশি সময় নেয়। এদিকে GOSUB দিয়ে লেখা প্রোগ্রামে কিন্তু e^x বের করার জন্য যে নির্দেশগুলি প্রয়োজন তা একবারই 200 সংখ্যক লাইন থেকে আরম্ভ করে 270 সংখ্যক লাইনে লেখা হয়েছে। এরপর প্রোগ্রামে যখনই x-এর কোনো একটি মানের জন্য e^x বের করার দরকার হয়েছে তখনই GOSUB 200 নির্দেশ ব্যবহার করে 280 সংখ্যক লাইনের নির্দেশের সাহায্যে যে GOSUB নির্দেশের জন্য এখানে এসেছে ঠিক তার পরের নির্দেশে ফিরে যাচ্ছে।

200 সংখ্যক লাইন থেকে 280 সংখ্যক লাইনের নির্দেশগুলি মূল প্রোগ্রাম থেকে আলাদা ভাবে একটি প্রোগ্রাম হিসেবে দেখা যেতে পারে, যদিও এটি একটি স্বয়ং সম্পূর্ণ প্রোগ্রাম হবে না। এর কারণ 200 সংখ্যক লাইন থেকে 280 সংখ্যক লাইনের মধ্যে কোথাও x-এর মান দেওয়া নেই। এই মান সব সময়েই মূল প্রোগ্রামটি থেকে সরবরাহ করা হয়। এছাড়া এখানে একটি RETURN নির্দেশ আছে যা কখনোই কোনো মূল প্রোগ্রামে থাকে না। এইজন্য এই নির্দেশগুলিকে একটি প্রোগ্রাম না বলে সাব-প্রোগ্রাম বা সাবরুটিন বলা হয়। এইসব সাব-প্রোগ্রামকে সব সময়েই অন্য প্রোগ্রামের উপর নির্ভর করতে হয়। GOSUB শব্দটি GO SUBROUTINE থেকে নেওয়া।

অনেক সময়ে একটি প্রোগ্রাম থেকে একাধিক সাব-রুটিনে যাওয়া হয়। সেক্ষেত্রে প্রয়োজনমত যে কোনো সাবরুটিনে গিয়ে সেখানকার নির্দেশগুলি পালন করে কমপিউটার আবার যে প্রোগ্রাম থেকে এসেছে সেই প্রোগ্রামে ফিরে আসে। এটাই নিয়ম। কোনো একটি প্রোগ্রাম থেকে সব সময়েই সাব-প্রোগ্রামগুলিতে এসে আবার সেই প্রেগ্রামেই ফিরে যেতে হয়। তবে সাধারণত বেসিকে সব

সাবরুটিনই একটি বড় প্রোগ্রামের অঙ্গ হিসেবেই লেখা হয়ে থাকে। যেভাবে উদাহরন 1 এখানে লেখা হয়েছে।

ON ... GOSUB এবং ON ... GOTO নির্দেশঃ

ON ... GOSUB-এর ব্যবহার একটি উদাহরণের সাহায্যে বোঝানো হচ্ছে। মনে করা যাক C-এর মান 1 হলে দশটি যে কোনো সংখ্যার গড় বের করতে হবে, আবার C-এর মান 2-এর ক্ষেত্রে দশটি সংখ্যার মধ্যে সবচেয়ে বৃহত্তম সংখ্যাটি বের করা হবে এবং C-এর মান 3 হলে দশটি সংখ্যার মধ্যে সবচেয়ে ক্ষুদ্রতমটি বের করা দরকার। অর্থাৎ C-এর বিভিন্ন মানের জন্য ভিন্ন ভিন্ন ধরনের কাজ করতে হবে। নীচে এই সমস্যার সমাধান বেসিক ভাষায় ON ... GOSUB ব্যবহার করে করা হচ্ছে।

উদাহরণ 1.

```
10 PRINT "This will find either average or maximum
or minimum of the numbers"
20 PRINT
30 PRINT "Select one of the following choices"
40 PRINT " (1) to find average"
50 PRINT " (2) to find maximum"
60 PRINT " (3) to find minimum"
70 PRINT
80 INPUT "Choice = "; C
90 ON C GOSUB 110, 200, 300
100 END
110 REM This part finds the average of ten numbers
120 SUM = 0
130 FOR I = 1 TO 10
140 INPUT "GIVE a number ="; N
150 SUM = SUM + N
160 NEXT I
170 AV = SUM/10
180 PRINT "The average of ten numbers ="; AV
190 RETURN
200 REM This part finds the maximum of ten numbers
```

```
210 INPUT "GIVE a number ="; N
220 MAX = N
230 FOR I = 2 TO 10
240 INPUT "give another number ="; N
250 IF MAX<N THEN MAX = N
260 NEXT I
270 PRINT "The maximum of ten numbers ="; MAX
280 RETURN
300 REM This part finds the minimum of ten numbers
310 INPUT "give a number ="; N
320 MIN = N
330 FOR I = 2 TO 10
340 INPUT "give another number ="; N
350 IF MIN > N THEN MIN = N
360 NEXT I
370 PRINT "The minimum of ten numbers ="; MIN
380 RETURN
```

এবারে দেখা যাক 90 সংখ্যক লাইনের নির্দেশঃ

ON C GOSUB 110, 200, 300

কিভাবে কাজ করবে। এই নির্দেশ অনুসারে C-এর মানের উপর নির্ভর করে 110, 200 বা 300 সংখ্যক লাইনের নির্দেশে গিয়ে যতক্ষণ পর্যন্ত না RETURN নির্দেশ পাবে ততক্ষণ সেখানকার নির্দেশগুলি পরপর করবে। RETURN নির্দেশ পেলেই এই নির্দেশের পরের লাইন সংখ্যায় অর্থাৎ 100 সংখ্যক লাইনে এসে সেখানকার নির্দেশ করবে। এখন এই ON ... GOSUB নির্দেশে GOSUB শব্দের পর মোট তিনটি লাইন সংখ্যা আছে। সেক্ষেত্রে C-এর মান 3-এর বেশি হওয়া সম্ভব নয়। C-এর মান 1, 2 বা 3 হতে পারে। C-এর মান 1 হলে GOSUB এর-পর প্রথম লাইন সংখ্যায় যাবে। এখানে প্রথম লাইন সংখ্যা 110। কাজেই সেখানে গিয়ে 110 থেকে শুরু করে 190 সংখ্যক লাইনের নির্দেশ পর্যন্ত পালন করবে। C-এর মান 2 এবং 3 হলে যথাক্রমে 200 এবং 300 লাইন সংখ্যায় গিয়ে সেখানকার নির্দেশ পালন করবে। কিন্তু প্রত্যেকটি ক্ষেত্রেই RETURN নির্দেশ পেলেই 100 সংখ্যক লাইনে ফিরে গিয়ে সেখানকার নির্দেশ অনুযায়ী থেমে যাবে। এই প্রোগ্রাম চালানো হলে C-এর যে কোনো একটি মানের জন্য যে কাজ করার কথা

কেবলমাত্র সেই কাজ করে কমপিউটার থেমে যাবে।

ON ... GOSUB নির্দেশ দেবার নিয়ম হল ON শব্দের পর একটি বেসিক সংখ্যার প্রকাশ এবং GOSUB-এর পর কতগুলি লাইন-সংখ্যার উল্লেখ। এখানে একটি লাইন-সংখ্যাকে অপর একটি লাইন-সংখ্যা থেকে আলাদা করার জন্য কমা চিহ্ন ব্যবহার করা হয়। সংখ্যার প্রকাশ বলতে বোঝানো হয় যে, এর মান সব সময়েই একটি অখণ্ড ধনাত্মক সংখ্যা হবে। অর্থাৎ তা 1, 2, 3, 4, 5, ..., K হওয়া সম্ভব। যদি K পর্যন্ত হয় তা হলে GOSUB-এর পরও K সংখ্যক লাইন-সংখ্যা থাকবে। ON-এর পর কেবলমাত্র কোনো সংখ্যা চলরাশিও থাকতে পারে। এই চলরাশির মানের উপর নির্ভর করে GOSUB-এর পরে যে-সব লাইন সংখ্যার উল্লেখ আছে কমপিউটার তার কোন একটি লাইন সংখ্যাতে যাবে। GOSUB-এরপর K সংখ্যক লাইন সংখ্যা থাকলে চলরাশিটির মান 1 থেকে আরম্ভ করে K পর্যন্ত হওয়া সম্ভব। কিন্তু চলরাশিটির মান কোনো কারণে 1-এর ছোট বা K-এর বেশি হলে কমপিউটার ON ... GOSUB নির্দেশের পরের নির্দেশ চলে যাবে। তবে অন্যান্য কিছু বেসিক কমপাইলার সেক্ষেত্রে ভুল হয়েছে বলে থেমে যাবে।

ON ... GOSUB-এর মত ON ... GOTO নির্দেশও প্রায় একই কাজ করবে। তফাত হলো GOSUB-এর ক্ষেত্রে কোনো লাইন সংখ্যায় গেলে সেখানকার নির্দেশ পালন করে আবার RETURN থাকাতে ON ... GOSUB নির্দেশের পরের নির্দেশে ফেরত আসে। কিন্তু ON ... GOTO নির্দেশের ক্ষেত্রে সেখানকার লাইন সংখ্যায় গিয়ে আর নিজের থেকে ON ... GOTO র পরের নির্দেশে ফেরত আসে না।

আগের উদাহরণটি এবারে ON ... GOTO দিয়ে দেখানো হচ্ছে।

উদাহরণ 2

```
10 PRINT "This program will find either average or maximum or minimum of ten numbers"
20 PRINT
30 PRINT "select one of the following choices"
40 PRINT " (1) to find average"
50 PRINT " (2) to find maximum"
60 PRINT " (3) to find minimum"
70 PRINT
80 INPUT "Choice = "; C
90 ON C GO TO 110, 200, 290
```

```
100 GO TO 380
110 REM This part finds the average of ten numbers
120 SUM = 0
130 FOR I = 1 TO 10
140 INPUT "Give a number ="; N
150 SUM = SUM + N
160 NEXT I
170 AV = SUM/10
180 PRINT "The average of ten numbers ="; AV
190 GO TO 100
200 REM This part find the maximum of ten numbers
210 INPUT "Give a number ="; N
220 MAX = N
230 FOR I = 2 TO 10
240 INPUT "give another number ="; N
250 IF MAX<N THEN MAX = N
260 NEXT I
270 PRINT "The maximum of ten numbers ="; MAX
280 GO TO 100
290 REM This part finds the minimum of ten numbers
300 INPUT "give a number ="; N
310 MIN = N
320 FOR I = 2 TO 10
330 INPUT "give another number ="; N
340 IF MIN>N THEN MIN = N
350 NEXT I
360 PRINT "The minimum of ten numbers ="; MIN
370 GO TO 100
380 END
```

ON ... GOSUB-এর ক্ষেত্রে C-এর মান যেমন 1, 2 বা 3 হওয়া সম্ভব ON ... GOTO-এর বেলাতেও C-এর মান ওই একই 1, 2 বা 3 হতে পারে। এক্ষেত্রেও C-এর মান 1, 2 বা 3 হলে কমপিউটার যথাক্রমে 110, 200 বা 290 সংখ্যক লাইন-সংখ্যায় যাবে। তবে দুটি নির্দেশের ক্ষেত্রে তফাত হল ON ... GOSUB-এর

বেলায় যেখানেই যাক না কেন সেখানকার RETURN নির্দেশের জন্য আবার ঠিক ON ... GOSUB-এর পরের নির্দেশে ফিরে আসবে। কিন্তু ON ... GOTO-এর বেলাতে যেখানেই যাবে সেখান থেকে পরপর নির্দেশগুলি করতে থাকবে। নিজের থেকে আর ON ... GOTO-এর পরের নির্দেশে ফিরে আসবে না। কাজেই সেখানে ফিরে আসার জন্য একটি নির্দেশের প্রয়োজন। উপরের উদাহরণে 190, 280 এবং 370 এই তিনটি লাইন সংখ্যাতেই GO TO 100 এই নির্দেশ দেওয়ার দরকার 90 সংখ্যক লাইনের ON ... GOTO নির্দেশের পরের নির্দেশে ফিরে যাওয়ার জন্য। ON ... GOSUB-এর ক্ষেত্রে কিন্তু তা নয়। কমপিউটার যে ON ... GOSUB নির্দেশ থেকে আসবে ঠিক তার পরের নির্দেশে নিজের থেকেই ফিরে যাবে। একই কাজ করার জন্য একটি প্রোগ্রামে GOSUB-এর মত বিভিন্ন জায়গাতে ON ... GOSUB নির্দেশ থাকা যেমন সম্ভব তেমনি আবার ON ... GOTO নির্দেশও লেখা যেতে পারে। সেক্ষেত্রে ON ... GOSUB-এর বেলায় ভিন্ন ভিন্ন ON ... GOSUB-এর জন্য ভিন্ন ভিন্ন লাইন-সংখ্যায় ফিরে যাবে। কিন্তু ON ... GOTO-এর ক্ষেত্রে সব ON ... GOTO-এর জন্য সেই-একই জায়গাতে ফিরে যাবে। কিছু সংখ্যক IF-THEN নির্দেশের জায়গায় অনেক সময়ে একটি ON ... GOTO নির্দেশের সাহায্যেই করা সম্ভব। একটি উদাহরণ দিয়ে ব্যাপারটা পরিষ্কার করা যেতে পারে। মনে করা যাক চলরাশি CODE-এর মানের উপর নির্ভর করে কত TAX হবে তা ঠিক করা হয়। নীচের টেবিলে কত CODE-এর জন্য কত TAX তা দেওয়া হলঃ

CODE	TAX
1	20%
2	25%
3	35%
4	45%

এখানে একজনের কত আয় এবং তার CODE কত তা জানার পর TAX বের করা সহজেই সম্ভব। নীচে এই TAX বেরকরার প্রোগ্রামটি দেওয়া হলঃ

উদাহরণ 3.

```
5 PRINT TAB(4); "EMP-NO", TAB(20); "CODE",
TAB(32); "EARNING", TAB(50); "TAX"
10 INPUT EARNING
20 IF EARNING = 0 THEN 150
```

```
30 INPUT EMP.NO, CODE
40 ON CODE GO TO 60, 80, 100, 120
50 GO TO 10
60 TAX = .2 * EARNING
70 GO TO 130
80 TAX = .25 * EARNING
90 GO TO 130
100 TAX = .35 * EARNING
110 GO TO 130
120 TAX = .45 * EARNING
130 PRINT TAB(6); EMP.NO, TAB(22); CODE,
TAB(31); EARNING, TAB(48); TAX
140 GO TO 10
150 END
```

এবারে উপরের প্রোগ্রামটি কি ভাবে কাজ করবে তা দেখা যাক। 5 সংখ্যক লাইনের কাজ কি ? পর্দায় নীচের লাইনটি ফুটিয়ে তোলা।

EMP-NO CODE EARNING TAX

এই লাইনে প্রথম E অক্ষরটি 4 সংখ্যক জায়গা থেকে শুরু হবে। এরপরের তিনটি শব্দের অক্ষর C, E এবং T যথাক্রমে 20, 32 এবং 50 সংখ্যক স্থানে ফুটে উঠবে। এটা সম্ভবপর হচ্ছে TAB(4), TAB(20), TAB(32) এবং TAB(50) থাকার জন্য। এরপর 10 সংখ্যক নির্দেশ অনুসারে EARNING চলরাশির মান পড়া হল। এই মান শূন্য হওয়ার অর্থ আর কোনো তথ্য নেই। কাজেই সেক্ষেত্রে কমপিউটার 150 সংখ্যক লাইনে গিয়ে থেমে যাবে। যদি শূন্য না হয় তবে EMP.NO এবং CODE কত তা পড়া হবে। CODE-এর মান 1, 2, 3 বা 4 হওয়া সম্ভব। এই মানের উপর নির্ভর করে 60, 80, 100 কিংবা 120 সংখ্যক লাইনের নির্দেশে গিয়ে সেখানকার নির্দেশ অনুসারে TAX বেরকরে কমপিউটার 130 সংখ্যক লাইনের নির্দেশে এসে যে চারটি চলরাশির উল্লেখ আছে তাদের মান ছাপাবে। কিন্তু কোথায় এদের মান ছাপা হবে ? একটি লাইনের কোথায় এদের মান ছাপা হবে তার জন্য TAB ফাংশনগুলি দেওয়া আছে। এই TAB ফাংশনগুলি এমনভাবে প্রদত্ত যাতে উপরের যে লাইনটি লেখা হয়েছে ঠিক তার নীচেই ওদের মানগুলিও লেখা হয়। এরপর 140 সংখ্যক লাইনে এসে আবার ঠিক আগের মতই 10 সংখ্যক লাইনে যাবে এবং আগের মতই নির্দেশগুলি পালন করতে

থাকবে। এইভাবে যখন আর কোনো তথ্য থাকবে না তখন EARNING-এর মান শূন্য দিলেই প্রোগ্রামটি থেমে যাবে। এখানে উল্লেখ করা যেতে পারে যে, 40 সংখ্যক লাইনের ON... GOTO নির্দেশের জায়গায় কিছু সংখ্যক IF-THEN নির্দেশ দিয়ে একই কাজ করা সম্ভব। নীচে IF-THEN নির্দেশের সাহায্যে এই ON-GOTO নির্দেশের কাজ দেখানো হচ্ছে।

```
40  IF  CODE  =  1   THEN  60
42  IF  CODE  =  2   THEN  80
44  IF  CODE  =  3   THEN  100
46  IF  CODE  =  4   THEN  120
```

অর্থাৎ এই প্রোগ্রামের জন্য একটি ON...GO TO নির্দেশের জায়গায় চারটি IF-THEN নির্দেশের প্রয়োজন।

গ্রাফিক্স (Graphics)

গ্রাফিক্স সম্বন্ধে কয়েকটি কথা ঃ

বেসিক ভাষায় ভিডিইউ-এর পর্দায় সাধারণ ভাবে দুটি প্রণালীতে লেখা সম্ভব। এক ঃ বেসিক বর্ণমালার সমস্ত অক্ষর লেখার প্রণালী অবলম্বনে—এক্ষেত্রে পর্দায় মোট 25 লাইন এবং প্রতি লাইনে 80টি অক্ষর পাওয়া যায়। এই প্রণালীতে যে কোনো রকমের বেসিক বর্ণমালার অক্ষর লেখা সম্ভব বলে একে আক্ষরিক প্রণালী (Text mode) বলা হয়ে থাকে এবং তা সাধারণ ভাবে সব বেসিক সংস্করণেই mode 0 হিসেবে চিহ্নিত করা হয়। এতক্ষণ পর্যন্ত PRINT নির্দেশের সাহায্যে পর্দায় যা লেখা হয়েছে তা এই প্রণালীর আওতায় আসে। দুইঃ এই প্রণালীতে কিছু সংখ্যক আলোক-বিন্দুর সাহায্যে বিভিন্ন ছবি পর্দায় ফুটিয়ে তোলা সম্ভব। এই প্রণালী গ্রাফিক্স প্রণালী (graphics mode) নামে অভিহিত। গ্রাফিক্স প্রণালী আবার দু রকমের হতে পারে।

1. মাঝারি রেজলিউশন গ্রাফিক্স—এই প্রণালীতে পর্দায় মোট 200 লাইন এবং প্রতি লাইনে মোট 320টি বিন্দু পাওয়া যায়। এই রকম গ্রাফিক্সের ক্ষেত্রে বিভিন্ন রকমের রং-এর ব্যবহার করা যায়। এই প্রণালীকে mode 1 বলা হয়।

2. উঁচু রেজলিউশন গ্রাফিক্স—এক্ষেত্রে পর্দায় মোট 200টি লাইন এবং প্রতি লাইনে 640টি বিন্দু থাকা সম্ভব। তবে এই ধরনের গ্রাফিক্সে কেবলমাত্র সাদা-কালো ছবি পাওয়া যায়। এই প্রণালীকে mode 2 নামে চিহ্নিত করা হয়।

কমপিউটারের পর্দায় বিভিন্ন ছবি ফুটিয়ে তোলার জন্য বিভিন্ন ধরনের নির্দেশের সাহায্য নেওয়া হয়। কিন্তু ছবি ফুটিয়ে তোলার জন্য কেবলমাত্র নির্দেশগুলিই যথেষ্ট নয়, কিছু বিশেষ ধরনের হার্ডওয়ারেরও প্রয়োজন। এই হার্ডওয়ারের অভাবেও ছবি ফুটিয়ে

তোলা সম্ভব হবে না। কাজেই পার্সোনাল কমপিউটারে ছবি ফুটিয়ে তোলার জন্য অবশ্যই ওই কমপিউটারে রং-বেরং গ্রাফিক্স মনিটর অ্যাডাপটর থাকা প্রয়োজন। আবার প্রিন্টারের সঙ্গে লাগানো কাগজের উপর এই ছবি ফুটিয়ে তোলার জন্য প্রিন্টারেও কিছু বিশেষ ব্যবস্থা থাকতে হবে। এখানে কেবলমাত্র পর্দায় রং-বেরং ছবি ফুটিয়ে তোলার জন্য যে-সব বিভিন্ন ধরনের নির্দেশ আছে সেগুলিই আলোচনা করা হবে। কাজেই যে-সব প্রোগ্রাম এখানে দেওয়া হয়েছে সেগুলি কমপিউটারে চালালে তবেই ওই ছবিগুলি দেখা যাবে।

টিভির মত এখানেও দু ধরনের ছবি হতে পারে—রং-বেরং এবং সাদা কালো। রং-বেরং গ্রাফিক্স মনিটর অ্যাডাপটর থাকলে 16টি বিভিন্ন রকমের রং ব্যবহার করা সম্ভব। এ ছাড়া পর্দায় ছবির বিপরীত (অর্থাৎ ছবিটি কালো পটভূমির উপর সাদা থাকলে এবারে সাদা পটভূমির উপর কালো হবে) এবং বিন্দুগুলি জ্বলছে নিভছে দেখানো যায়। মনোক্রোম বা সাদা-কালো মনিটর থাকলে ছবির বিপরীত এবং বিন্দুগুলি জ্বলছে নিভছে করা চলে। কিন্তু সাদা কালো ছাড়া অন্য কোনো রং-এর ছবি পাওয়া যাবে না।

গ্রাফিক্সে যে আলোক-বিন্দুর সাহায্যে ছবি পাওয়া যায় তাদের পিক্সেল বলে। পিক্সেল (Pixels) শব্দটি Picture elements থেকে গৃহীত। পর্দায় বিন্দুগুলিকে নির্দিষ্ট করার জন্য স্থানাঙ্কের সাহায্য নেওয়া হয়। মাঝারি রেজলিউশন গ্রাফিক্সের ক্ষেত্রে মোট 320 × 200 = 64000 বিন্দু পাওয়া যায়। এখানে প্রথম লাইনকে শূন্য দিয়ে চিহ্নিত করায় শেষ লাইনকে 199 ধরা হয়। একটি লাইনে বিন্দুও শূন্য থেকে আরম্ভ করে 319 পর্যন্ত হতে পারে এবং প্রথম লাইনের প্রথম বিন্দু ও শেষ লাইনের শেষ বিন্দুর স্থানাঙ্ক যথাক্রমে (0, 0) এবং (319, 199) দিয়ে বোঝানো হয়ে থাকে।

এবারে গ্রাফিক্সের জন্য যে-সব নির্দেশ ব্যবহার করা হয় তা আলোচনা করা যাক।

SCREEN নির্দেশঃ

এই নির্দেশ দেওয়ার নিয়ম হল SCREEN শব্দের পর এক বা একাধিক ফাঁকা জায়গা এবং তারপর একটি সংখ্যা। এই সংখ্যা 0, 1 বা 2 যে কোনোটিই হওয়া সম্ভব।

উদাহরণ 1.

```
10 SCREEN 1
```

এই নির্দেশে সংখ্যাটি 1 থাকাতে বোঝা যাবে যে, এবারে

মাঝারি রেজলিউশনের গ্রাফিক্সের ব্যবহার সম্ভব। সংখ্যাটি 2 হলে উঁচু রেজলিউশন গ্রাফিক্স এবং 0 হলে আক্ষরিক প্রণালী অর্থাৎ সাধারণ পদ্ধতি বোঝাবে। তবে বেসিকের কোনো প্রোগ্রামের শুরুতেই SCREEN 0 লেখার দরকার হয় না। তবে SCREEN 1 এবং SCREEN 2 পদ্ধতিতে কাজ করার পর আক্ষরিক প্রণালীতে ফিরে আসার প্রয়োজনে SCREEN 0 ব্যবহার করা হয়। কিন্তু প্রোগ্রামের প্রথমে SCREEN নির্দেশ না লেখা হলে ধরে নেওয়া হবে যে, কেবলমাত্র সাধারণ পদ্ধতিতেই পর্দায় লেখা হবে। অনেক সময়ে ছবির সঙ্গে সঙ্গে অক্ষরও লেখার প্রয়োজন হতে পারে। SCREEN নির্দেশের পর বেসিকের যে-সব নির্দেশ এর আগে আলোচনা করা হয়েছে অক্ষরের বেলায় তা সবই ব্যবহার করা সম্ভব। তবে অক্ষরগুলি ছাপালে সাধারণ যে আকারে তা পর্দায় ফুটে ওঠে এখানে তার চেয়ে বড় হবে। গ্রাফিক্সের ব্যবহার শুরু হলে কোনো অক্ষর ছাপানোর জন্য $8 \times 8 = 64$টি বিন্দুর প্রয়োজন। আগেই বলা হয়েছে যে, মাঝারি রেজলিউশন গ্রাফিক্সে এক লাইনে মোট 320টি বিন্দু থাকে। কাজেই এক্ষেত্রে এক লাইনে মোট $320/8 = 40$টি অক্ষর পর্দায় লেখা যায়। কিন্তু গ্রাফিক্স ব্যবহারের আগে এক লাইনে 80টি পর্যন্ত অক্ষর লেখা যায়। কাজেই গ্রাফিক্সে অক্ষরগুলি আকারে কিছুটা বড় হবে।

মাঝারি রেজলিউশন গ্রাফিক্সে এক লাইনে 40টি অক্ষর লেখা যায় বলে একে WIDTH 40 বলা হয়, যদিও এক্ষেত্রে WIDTH 40 নির্দেশ আর আলাদা করে দেওয়ার প্রয়োজন হয় না। কিন্তু গ্রাফিক্স থেকে বেসিকের আক্ষরিক রেজলিউশনে ফিরে যাওয়ার প্রয়োজনে নীচের দুটি নির্দেশ দিতে হবে।

উদাহরণ 2.

```
500 SCREEN 0
510 WIDTH 80
```

এখানে 500 সংখ্যক লাইনের নির্দেশ অনুসারে কমপিউটার আক্ষরিক রেজলিউশনে ফিরে যাবে। কিন্তু 510 সংখ্যক লাইনের নির্দেশ না থাকলে এক লাইনে 40টি অক্ষরই থাকবে। আবার WIDTH 80 নির্দেশ প্রথমে পর্দার সব লেখা মুছে দিয়ে পর্দায় 80টি অক্ষর লেখার অবস্থায় ফিরিয়ে আনবে। এখানে মনে রাখা ভাল যে, পর্দার জন্য কেবলমাত্র WIDTH 40 এবং WIDTH 80 হওয়া সম্ভব।

COLOR নির্দেশঃ

রং বেরংয়ের ছবি পর্দায় ফোটানের জন্য এই নির্দেশের প্রয়োজন।

নির্দেশটি লেখার নিয়ম হলঃ

100 COLOR পটভূমি, প্যালিট

পটভূমির রং কি হবে তা বোঝানোর জন্য 0 থেকে 15 পর্যন্ত যে কোনো একটি সংখ্যার সাহায্য নেওয়া হয়। কোন সংখ্যা কি রং বোঝাবে তা নীচে দেওয়া হলঃ

সংখ্যা	রং	সংখ্যা	রং
0	কালো	8	রুপালী
1	নীল	9	ফিকে নীল
2	সবুজ	10	ফিকে সবুজ
3	মাঝারি নীল	11	ফিকে মাঝারি নীল
4	লাল	12	ফিকে লাল
5	বেগুনী-লাল	13	ফিকে বেগুনী-লাল
6	বাদামী	14	হলুদ
7	সাদা	15	ধবধবে সাদা

মাঝারি রেজলিউশন গ্রাফিক্সের পটভূমির রং

এই নির্দেশ ব্যবহার করলে একেবারে পুরো পর্দার পটভূমির রং সংখ্যাটির উপর নির্ভর করবে। যদি পটভূমির জন্য কোনো সংখ্যা ব্যবহার করা না হয় সেক্ষেত্রে মাঝারি রেজলিউশন গ্রাফিক্সের বেলায় পটভূমির রং কালো ধরা হবে।

প্যালিট (Palette) বলতে একজন চিত্রকর যে জিনিষের উপর রং মেশান তাকে বোঝান হয়ে থাকে।

এখানে প্যালিট-এর ক্ষেত্রে কেবলমাত্র 0 এবং 1 সংখ্যা দুটি ব্যবহার করা সম্ভব। অবশ্য প্যালিটের উল্লেখ করা না হলে 1 ধরে নেওয়া হয়। আবার প্যালিট 0 হলে যে কোনো চারটি রং-এর ছবি পাওয়া যাবে– পটভূমির, সবুজ, লাল, ও বাদামী। প্যালিট 1-এ যে চারটি রংয়ের ছবি পাওয়া যাবে তা হল পটভূমির, মাঝারি নীল, বেগুনী-লাল এবং সাদা। নীচের টেবিলের সাহায্যে দেখানো হচ্ছে কোন ধরনের প্যালিটের রংয়ের জন্য কোন সংখ্যা ব্যবহার করলে ছবির কি রং হবে।

রং	প্যালিট 0	প্যালিট 1
0	পটভূমির রং	পটভূমির রং
1	সবুজ	মাঝারি নীল
2	লাল	বেগুনী-লাল
3	বাদামী	সাদা

COLOR নির্দেশে প্যালিটের উল্লেখ না থাকলে SCREEN 1 নির্দেশের জন্য তা 1 ধরা হয়। উপরের টেবিলে রং-এর জন্য যে সংখ্যার উল্লেখ করা হয়েছে সেই সংখ্যা কোথায় কি ভাবে উল্লেখ করতে হবে তা নীচের উদাহরণগুলিতে এবং PSET ও LINE নির্দেশ আলোচনার সময়ে দেখানো হবে। PRINT নির্দেশের সাহায্যে কোনো অক্ষর ছাপানো হলে রংয়ের সংখ্যা সব সময়েই 3 ধরা হয় তা প্যালিট 0 বা 1 যাই হোক। কোন নির্দেশের জন্য ছবির রং এবং পটভূমির রং কি হবে নীচের উদাহরণগুলির সাহায্যে তা দেখানো হচ্ছে।

উদাহরণ 3.

```
100 COLOR 2, 1
110 LOCATE 10, 10
120 PRINT "GREEN ON WHITE"
```

100 সংখ্যক লাইনের নির্দেশে প্রথম সংখ্যা 2 থাকাতে পটভূমির রং সবুজ নেওয়া হয়েছে। এরপরের সংখ্যা 1 হওয়ায় প্যালিট-এর সংখ্যা 1। কাজেই PRINT নির্দেশের সাহায্যে যে অক্ষরগুলি পর্দায় ফোটানো হচ্ছে তাদের রং হবে সাদা। কারণ এখানে প্যালিট 1 এবং PRINT-এর জন্য রংয়ের সংখ্যা 3 ধরা হয়েছে। অক্ষরগুলি পর্দায় কোন লাইনে এবং সেই লাইনের কোন স্থান থেকে লেখা হবে তা বোঝাতে LOCATE নির্দেশের ব্যবহার। LOCATE নির্দেশে 10, 10 থাকায় 10 সংখ্যক লাইনের 10 সংখ্যক জায়গা থেকে নীচের অক্ষরগুলি সবুজ পটভূমির উপর সাদা রংয়ে লেখা হবে।

GREEN ON WHITE

উদাহরণ 4.

```
100 COLOR 2, 0
110 LOCATE 10, 10
120 PRINT "GREEN ON BROWN"
```

এবারে সবুজ পটভূমির উপর বাদামী রংয়ে লেখা হবে

GREEN ON BROWN

100 সংখ্যক লাইনের নির্দেশে প্যালিটের সংখ্যা 0 এবং PRINT নির্দেশের জন্য রংয়ের সংখ্যা হবে 3। এবারে টেবিল থেকে দেখা যাচ্ছে প্যালিট 0-তে 3 সংখ্যার রং হল বাদামী।

PSET নির্দেশঃ

মাঝারি রেজলিউশন গ্রাফিক্সে 64000 বিন্দুর যে কোনো একটি

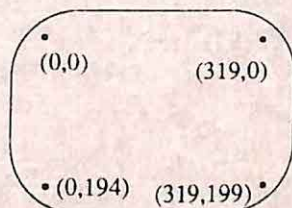
বিন্দুকে আলোকিত করার জন্য এই নির্দেশের প্রয়োজন। এই নির্দেশ দেওয়ার নিয়ম হলঃ

PSET(x, y)

এখানে x, y-এর সাহায্যে বিন্দুটির স্থানাঙ্ক বোঝানো হচ্ছে। নীচের উদাহরণের সাহায্যে পর্দায় কিছু সংখ্যক বিন্দুকে আলোকিত করার জন্য যে-সব নির্দেশের প্রয়োজন তা দেখানো হল।

উদাহরণ 5.

```
10 SCREEN 1
20 CLS
30 KEY OFF
40 PSET (0, 0)
50 PSET (319, 0)
60 PSET (0, 199)
70 PSET (319, 199)
80 SCREEN 0
90 WIDTH 80
100 END
```

উপরের উদাহরণে 10 সংখ্যক লাইনের নির্দেশে 1 সংখ্যার ব্যবহারের সাহায্যে বোঝানো হচ্ছে যে, এবারে মাঝারি রেজলিউশনের গ্রাফিক্স করা সম্ভব। এরপরের নির্দেশ দুটির সাহায্যে পর্দাকে একেবারে মুছে দেওয়া হল অর্থাৎ পর্দা একেবারে পরিষ্কার দেখাবে। KEY OFF ব্যবহার করায় বেসিকে পর্দার একেবারে শেষ লাইনে যে-সব ফাংশন কী-এর উল্লেখ থাকতো, তাও মুছে যাবে। এরপর 40 থেকে 70 সংখ্যক লাইনের নির্দেশের জন্য পর্দায় যে চারটি বিন্দু ফুটে উঠবে তা নীচে দেখানো হচ্ছে।

80 সংখ্যক লাইনের নির্দেশ অনুসারে কমপিউটার আবার গ্রাফিক্স থেকে বেসিকের সাধারণ প্রণালীতে ফিরে যাবে এবং এই পদ্ধতিতে এক লাইনে যাতে 80টি অক্ষর লেখা সম্ভব হয় তার জন্য 90 সংখ্যক লাইনের নির্দেশ প্রয়োজন।

অনেক সময়ে যে বিন্দুটি পর্দায় লেখা হল তারপর অন্য একটি বিন্দু বোঝাতে STEP-এর সাহায্য নেওয়া হয়।

উদাহরণ 6.

```
60 PSET (10, 80)
70 PSET STEP (5, –10)
```

60 সংখ্যক লাইনের নির্দেশ অনুসারে পর্দায় (10, 80) স্থানাঙ্কে একটি বিন্দু বসবে। এর পরের নির্দেশে STEP থাকায় এবারের বিন্দুটি (10 + 5, 80 – 10) অর্থাৎ (15, 70) স্থানাঙ্কে বসবে।

একই লাইনে 2টি করে পিক্সেল তফাতে 10টি বিন্দু বসানোর জন্য নীচের প্রোগ্রামটি লেখা যেতে পারে।

উদাহরণ 7.

```
100 PSET (50, 40)
120 FOR I = 1 TO 9
130 PSET STEP (2, 0)
140 NEXT I
150 END
```

নীচের প্রোগ্রামটির সাহায্যে একটি বর্গক্ষেত্র অঞ্চলের প্রতি দ্বিতীয় বিন্দুকে পর্দায় ফুটিয়ে তোলা হচ্ছে।

উদাহরণ 8.

```
70 SCREEN 1
80 CLS
90 KEY OFF
100 FOR I = 70 TO 120 STEP 2
110 PSET (170, I)
120 FOR J = 1 TO 25
130 PSET STEP (2, 0)
140 NEXT J
150 NEXT I
160 SCREEN 0
170 WIDTH 80
180 END
```

এই প্রোগ্রামের জন্য যে ছবি পাওয়া যাবে তা কমপিউটারে প্রোগ্রামটি চালিয়েই দেখে নেওয়া যেতে পারে। তবে ছবিটি পর্দায় ধরে রাখতে হলে নীচের নির্দেশটি দেওয়ার প্রয়োজন। এই নির্দেশ

কেন প্রয়োজন তা উদাহরণ 15তে আলোচনা করা হয়েছে।

155 IF INKEY $ ="" THEN 155

উপরের উদাহরণে 110 লাইনের নির্দেশ অনুসারে প্রথমবার I-এর মান 70 হওয়ায় লাইন সংখ্যা 70-এর 170 সংখ্যক স্থানে একটি বিন্দু বসবে। এরপর ওই একই লাইনে 170-এর পর থেকে প্রতি দ্বিতীয় স্থানে একটি করে বিন্দু বসতে থাকবে এবং এরকম 25টি বিন্দু পাওয়া যাবে। প্রথম বিন্দুটি ধরে সর্বমোট বিন্দুর সংখ্যা 26। এরপর আবার এই লাইনের মতই 72 লাইন সংখ্যাতেও 26টি বিন্দু বসবে এবং এক্ষেত্রেও দুটি বিন্দুর মধ্যে একটি ফাঁকা স্থান থাকবে। এইভাবে মোট 26টি লাইন এবং প্রতিটি লাইনে 26টি বিন্দু পর্দায় দেখা যাবে।

প্রত্যেকটি বিন্দুকে কোনো একটি রংয়ে প্রকাশ করা সম্ভব। এর জন্য যে নির্দেশ দেওয়া হয়ে থাকে তা হলঃ

PSET (x, y), সংখ্যা

এই সংখ্যা 0, 1, 2 বা 3 হওয়া সম্ভব। কোন্ প্যালিটের কোন সংখ্যার জন্য কি রং হবে তা COLOR নির্দেশ আলোচনা করার সময়েই বলা হয়েছে।

উদাহরণ 9.

```
100 COLOR 2, 1
110 PSET (170, 70), 2
```

এখানে COLOR নির্দেশে প্রথম সংখ্যাটি 2 থাকাতে পর্দায় পটভূমির রং হবে সবুজ। দ্বিতীয় সংখ্যাটি 1 দিয়ে প্যালিটের সংখ্যা, বোঝানো হচ্ছে। এরপর 110 সংখ্যক লাইনের নির্দেশে সংখ্যা 2 থাকায় (170, 70) স্থানাঙ্কে যে বিন্দুটি পর্দায় দেখা যাবে তার রং হবে বেগুনী-লাল। কিন্তু 110-এ 2-এর জায়গায় 1 থাকলে হোত মাঝারি নীল। এই লাইনে এরপর প্রতিটি দ্বিতীয় বিন্দুকে এই বেগুনী-লাল রং দিয়ে পর্দায় ফোটানোর প্রয়োজন হলে নীচের নির্দেশগুলি লিখতে হবে।

উদাহরণ 10.

```
80 SCREEN 1
90 CLS : KEY OFF
100 COLOR 2, 1
110 PSET (170, 70), 2
120 FOR I = 1 TO 74
130 PSET STEP (2, 0), 2
140 NEXT I
```

```
150 SCREEN 0
160 WIDTH 80
170 END
```

120 সংখ্যক লাইনের নির্দেশে 74 লেখার কারণ কি ? এর কারণ 170 থেকে আরম্ভ করে 319 পর্যন্ত হবে 149 এবং প্রতিটি দ্বিতীয় বিন্দু পর্দায় লেখা হচ্ছে বলে 149/2 = 74.5 অর্থাৎ 74টি বিন্দু এক্ষেত্রে পাওয়া সম্ভব।

PSET নির্দেশ লেখার নিয়ম নীচে দেখানো হচ্ছে ঃ

PSET (x, y), c

অথবা

PSET STEP (a, b), c

এখানে x, y, a, b and c সংখ্যা। এরা ভগ্নাংশ হলে এদের পূর্ণ সংখ্যায় রূপান্তর ঘটানো হয়। x এবং y পর্দায় বিন্দুটির স্থানাঙ্ক নির্দেশ করে। আবার a এবং b-এর সাহায্যে পরের বিন্দুর স্থানাঙ্ক পাওয়া সম্ভব। প্রথম বিন্দুর স্থানাঙ্ক (x, y) হলে পরের বিন্দুর স্থানাঙ্ক হবে (x + a, y + b)। কিন্তু কোন রং-এ বিন্দুটি হবে তা বোঝানোর জন্য c সংখ্যাটি ব্যবহার করা হয়।

1. c শূন্য দিয়ে পটভূমির রং বোঝানো হয়।

2. c উল্লেখ না থাকলে যে প্যালিট তখন ব্যবহৃত হচ্ছে 3 সংখ্যকের যে রং তার সেই রং হবে। অর্থাৎ প্যালিট 0 হলে বাদামী এবং প্যালিট 1 হলে হবে সাদা রং।

3. c-এর মান 0, 1, 2 বা 3 হওয়া সম্ভব। c র মান 0-এর চেয়ে ছোট বা 255-এর থেকে বড় হলে ভুল হবে। কিন্তু c-এর মান 3 এর থেকে বড় হলে কমপিউটার 3 ধরে নেবে।

4. x এবং y-এর মান যথাক্রমে 0 থেকে 319 এবং 0 থেকে 199 হতে পারে।

PRESET নির্দেশঃ

এই নির্দেশও PSET নির্দেশের মতই কাজ করে। তবে এই নির্দেশ দুটির তফাত কেবলমাত্র একটি ব্যাপারে। রংয়ের কোনো উল্লেখ না থাকলে PSET-এর বেলায় পুরোভূমির রং ব্যবহার করা হয় এবং PRESET-এর ক্ষেত্রে পটভূমির রং নেওয়া হয়। রং-এর উল্লেখ না থাকার অর্থ হবে নির্দেশগুলি নীচের মত লেখা থাকবে।

PRESET (x, y) এবং PSET (x, y)

এখানে c-এর কোনো উল্লেখই থাকছে না।

PRESET নির্দেশ লেখার নিয়মও একেবারে PSET-এর মতই।

LINE নির্দেশঃ

এই নির্দেশের সাহায্যে দুটি স্থানাঙ্কের মধ্যে একটি সরলরেখা, একটি বাক্সের চারপাশ বা সম্পূর্ণ বাক্স আঁকা সম্ভব। পর্দায় বাক্স আঁকার অর্থ হবে একটি আয়তক্ষেত্র আঁকা। কাজেই বাক্সের চারপাশ বলতে একটি আয়তক্ষেত্রের চার বাহু বোঝানো হচ্ছে এবং সেক্ষেত্রে সম্পূর্ণ বাক্স আঁকার অর্থ হবে আয়তক্ষেত্রটি রং দিয়ে ভরে দেওয়া। এই LINE নির্দেশ দেওয়ার সবচেয়ে সহজ উপায় হলঃ

LINE (x1, y1) – (x2, y2)

উপরের নির্দেশ অনুসারে (x1, y1) এবং (x2, y2) এই দুটি স্থানাঙ্কের মধ্যে কমপিউটার একটি সরলরেখা আঁকবে। রংয়ের উল্লেখ না থাকায় এক্ষেত্রে বর্তমান প্যালিট সংখ্যার 3-এর জন্য যে রং সেই রংই হবে। অর্থাৎ বর্তমান প্যালিট 0 হলে বাদামী রং এবং 1-এর ক্ষেত্রে সাদা। যদি অন্য কোনো রং প্রয়োজন হয় তাহলে নির্দেশের ধরন হবে –

LINE (x1, y1) – (x2, y2), c

c-এর মান 0, 1, 2 বা 3 হতে পারে। বর্তমান প্যালিটে c-এর মানের উপর নির্ভর করে সরলরেখাটি সেই রংয়ে আঁকা হবে।

উদাহরণ 11.

```
100 LINE (15, 50) – (175, 50)
```

এই নির্দেশের ফলে স্থানাংক (15, 50) থেকে (175, 50) পর্যন্ত একটি সরলরেখা আঁকা হবে। এখানে পর্দায় 50 সংখ্যক লাইনের 15 সংখ্যক বিন্দু থেকে 175 সংখ্যক বিন্দু পর্যন্ত সরলরেখাটি হবে –

(15, 50) ———— (175, 50)

উদাহরণ 12.

```
100 LINE (15, 50) – (15, 150)
```

এবারেও একটি সরলরেখাই আঁকা হবে তবে এক্ষেত্রে সরলরেখাটি পর্দায় 50 সংখ্যক লাইনের 15 সংখ্যক বিন্দু থেকে আরম্ভ করে 150 সংখ্যক লাইনের 15 বিন্দু পর্যন্ত হবে।

(15, 50)

(15, 150)

উদাহরণ 13.

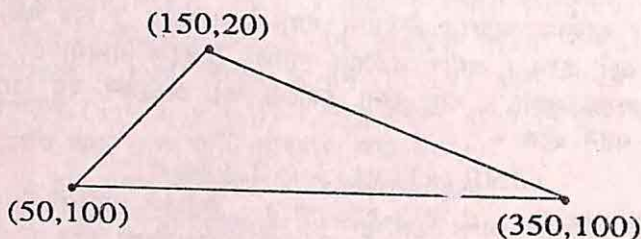
```
100 LINE (15, 50) – (250, 150)
```

এবারে কিন্তু সরলরেখাটি হবে নীচের মত।

(15, 50) ——— (250, 150)

উদাহরণ 14.

```
100 PSET (150, 20)
110 LINE –(350, 100)
120 LINE –(50, 100)
130 LINE –(150, 20)
```

এবারে তিনটি সরলরেখা আঁকা হবে এবং এই সরলরেখাগুলি নীচে দেখানো হচ্ছে।

উপরের উদাহরণে 100 সংখ্যক নির্দেশ অনুসারে (150, 20) স্থানাঙ্কে একটি বিন্দু বসবে। এরপর 110 সংখ্যক লাইনের নির্দেশের জন্য (150, 20) এবং (350, 100)-এর মধ্যে একটি সরলরেখা আঁকা হবে। এরপরের নির্দেশ অনুসারে (350, 100) এবং (50, 100)-এর মধ্যে অন্য একটি সরলরেখা পাওয়া যাবে। পরবর্তী নির্দেশের জন্য এই (50, 100) স্থানাঙ্ক থেকে (150, 20) স্থানাঙ্কের মধ্যে আবার একটি সরলরেখা আঁকা হবে। এখানে উল্লেখ করা প্রয়োজন যে, LINE নির্দেশে কেবলমাত্র একটি স্থানাঙ্কই দেখানো হচ্ছে। এক্ষেত্রে আগের স্থানাঙ্ক থেকে ওই স্থানাঙ্ক ধরে নেওয়া হবে। 110 সংখ্যক লাইনের নির্দেশ অনুসারে সরলরেখাটি আঁকার সময়ে আগের নির্দেশের স্থানাঙ্ক থেকে সরলরেখাটি আঁকা হবে। সেরকমই 120 সংখ্যক লাইনের নির্দেশ পালন করার সময় 110 সংখ্যক লাইনে যে স্থানাঙ্কের উল্লেখ আছে সেখান থেকে কমপিউটার সরলরেখাটি আঁকবে।

PSET নির্দেশের মত এক্ষেত্রেও রংয়ের ব্যবহার সম্ভব। উপরের উদাহরণে ত্রিভুজের তিনটি বাহুই রং দিয়ে আঁকতে হলে

নীচের প্রোগ্রামের সাহায্যে তা করা সম্ভব।

উদাহরণ 15.

```
10 SCREEN 1
20 CLS
30 KEY OFF
40 COLOR 2, 1
50 PSET (150, 20), 2
60 LINE –(350, 100), 2
70 LINE –(50, 100), 2
80 LINE –(150, 20), 2
85 IF INKEYS = " " THEN 85
90 SCREEN 0
100 WIDTH 80
110 END
```

উপরের প্রোগ্রাম চালানো হলে পর্দায় সবুজ পটভূমিতে বেগুনী-লালে ত্রিভুজের তিনটি বাহু পাওয়া যাবে। 40 সংখ্যক লাইনে COLOR নির্দেশে 2 থাকায় পটভূমির রং হবে সবুজ। এখানে 1 দিয়ে প্যালিটের সংখ্যা বোঝানো হচ্ছে। এরপর 50, 60, 70 এবং 80 সংখ্যক লাইনের নির্দেশে রংয়ের সংখ্যা 2 থাকাতে ত্রিভুজের বাহু তিনটির রং বেগুনী-লাল হবে। এই উদাহরণে 85 সংখ্যক লাইনের নির্দেশ সম্বন্ধে আলোচনা করা দরকার। এই নির্দেশের প্রয়োজন কি ? প্রোগ্রামে এই নির্দেশটি না থাকলে পর্দায় ছবিটি আঁকার সঙ্গে সঙ্গেই মিলিয়ে যাবে। কিন্তু এই নির্দেশটি থাকায় যতক্ষণ পর্যন্ত কোনো অক্ষর, চিহ্ন বা সংখ্যা কী-বোর্ডের মাধ্যমে দেওয়া না হচ্ছে ততক্ষণ পর্দায় ছবিটি ধরা থাকবে। কিন্তু কী-বোর্ডের মাধ্যমে কোনো কিছু টাইপ করলেই ছবিটি মিলিয়ে গিয়ে আবার পর্দায় OK শব্দটি ফুটে উঠবে। INKEY$ একটি সারি চলরাশির নাম। এই নির্দেশে বলা হচ্ছে, INKEY$ চলরাশির মান কিছু না দেওয়া পর্যন্ত অপেক্ষা করতে হবে। এখানে সমান চিহ্নের ডানপাশে দুটি উদ্ধৃতি চিহ্ন পাশাপাশি রয়েছে। এই দুটির মধ্যে কোনো ফাঁকা জায়গা না থাকার অর্থ হচ্ছে INKEY$ এর মান যতক্ষণ পর্যন্ত কী-বোর্ডের মাধ্যমে কিছু দেওয়া না হচ্ছে ততক্ষণ এই নির্দেশের শর্তটি সত্য থাকছে। কাজেই আবার 85 সংখ্যক লাইনে যাচ্ছে অর্থাৎ এই লাইনে এসে এখানকার শর্তটি সত্য কিনা তা আবার পরীক্ষা করে দেখছে। যে মুহূর্তে কী-বোর্ডের মাধ্যমে কিছু টাইপ করা হচ্ছে শর্তটি মিথ্যে হয়ে যাবে এবং তখন পরের নির্দেশে

চলে যাবে। LINE নির্দেশেও STEP-এর ব্যবহার সম্ভব।

উদাহরণ 16.

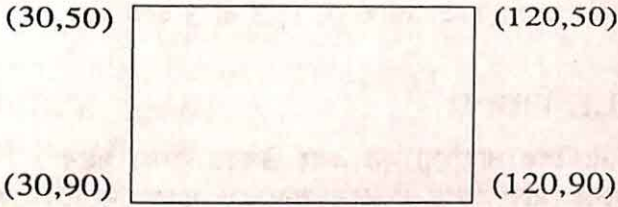
```
100 SCREEN 1
110 CLS
120 KEY OFF
130 COLOR 0, 1
140 FOR A = –50 TO 100 STEP 10
150 LINE (180, 90) –STEP (90, A), 3
160 NEXT A
165 IF INKEY$ =" " THEN 165
170 SCREEN 0
180 WIDTH 80
190 END
```

উপরের উদাহরণ কি ছবি ছাপাবে ? COLOR নির্দেশে 0 থাকায় এই ছবির পটভূমি হবে কালো। এই নির্দেশে এরপরের সংখ্যা প্যালিটের সংখ্যা 1 নির্দিষ্ট করছে FOR -NEXT আবর্তের মধ্যে LINE নির্দেশ থাকায় এই নির্দেশটি কয়েকবার করবে এবং এরফলে কিছু সংখ্যক সরলরেখা আঁকা হবে। এইসব সরলরেখার রং হবে সাদা। প্যালিট 1-এ 3 সংখ্যা সাদা রং বোঝায়। প্রথম সরলরেখাটি (180, 90) স্থানাঙ্কের বিন্দু থেকে আরম্ভ করে (180 + 90, 90 – 50) অর্থাৎ (270, 40) স্থানাঙ্কের বিন্দু পর্যন্ত হবে। এরপরের সরলরেখাটি (180, 90) থেকে (270, 90 – 40) অর্থাৎ (270, 50) পর্যন্ত বিস্তৃত। এইভাবে প্রত্যেকটি সরলরেখাই (180, 90) বিন্দু থেকে শুরু হবে এবং শেষ হওয়ার ক্ষেত্রে x-স্থানাঙ্ক 270-ই থাকবে কিন্তু y-স্থানাঙ্ক 40 থেকে আরম্ভ করে যাবে 190 পর্যন্ত এবং দুটি সরলরেখার মধ্যে y-এর ব্যবধান থাকবে 10। ছবিটি দেখতে হলে এই প্রোগ্রামটি কমপিউটারে চালাতে হবে।

এতক্ষণ পর্যন্ত LINE নির্দেশের সাহায্যে কেবলমাত্র একটি সরলরেখাই আঁকা হয়েছে। এবারে LINE নির্দেশ দিয়ে একটি বাক্স আঁকা হবে।

উদাহরণ 17.

```
100 LINE (30, 50) – (120, 90), B
```

উপরের নির্দেশের জন্য পর্দায় একটি বাক্সের ছবি ফুটে উঠবে যার দুই কৌণিক বিন্দুর স্থানাঙ্ক হবে (30, 50) এবং (120, 90)।

এই বাক্সের চারদিকে রং দেওয়ার প্রয়োজন হলে LINE নির্দেশেই তা দেওয়া যায়।

উদাহরণ 18.

100 LINE (30, 50) – (120, 90), 2, B

এবারে B-এর আগের সংখ্যাটি দিয়ে রং বোঝানো হচ্ছে। পুরো বাক্সই রং দিয়ে ভর্তি করতে হলে নীচের নির্দেশের সাহায্যেই তা করা সম্ভব।

উদাহরণ 19.

100 LINE (30, 50) – (120, 90), 2, BF

উপরের নির্দেশগুলিতেও STEP ব্যবহার করা যায়।

উদাহরণ 20.

100 LINE (30, 50) -STEP (90, 40), 2, BF

এই নির্দেশ এবং উদাহরণ 19 এ যে নির্দেশ দেওয়া হয়েছে তা একই কাজ করবে।

LINE নির্দেশের সাহায্যে সরলরেখা, বাক্স অর্থাৎ একটি আয়তক্ষেত্রের চারদিকের রেখা এবং ভর্তি-বাক্স করা সম্ভব। LINE নির্দেশ দেওয়ার নিয়ম হলঃ

LINE (x1, y1) – (x2, y2), c, f

f-এর উল্লেখ না থাকলে এই নির্দেশে একটি সরলরেখা পাওয়া যাবে যার দুই বিন্দুর স্থানাঙ্ক হবে (x1, y1) এবং (x2, y2)।

কিন্তু f = B হলে এই নির্দেশে একটি আয়তক্ষেত্রের চার দিক ফুটে উঠবে। এটির দুই কৌণিক বিন্দুর স্থানাঙ্ক হবে (x1, y1) এবং (x2, y2)।

f = BF হলে আয়তক্ষেত্রটি ভর্তি হবে।

(x1, y1) দেওয়া না থাকলে আগে যে বিন্দুর স্থানাঙ্ক দেওয়া থাকবে সেই বিন্দু এবং (x2, y2) বিন্দু নেওয়া হবে। (x2, y2)-এর জায়গাতে STEP-এর উল্লেখ থাকা সম্ভব। c-এর উল্লেখ না থাকলে বর্তমানে প্যালিটের 3 সংখ্যার জন্য যে রং তা ব্যবহার করা

হবে। কিন্তু যদি c-এর উল্লেখ থাকে তাহলে তা অবশ্যই একটি সঠিক সংখ্যা হওয়া চাই অর্থাৎ 0, 1, 2 বা 3 হতে হবে।

CIRCLE নির্দেশঃ

এই নির্দেশের সাহায্যে বৃত্ত এবং উপবৃত্ত আঁকা সম্ভব। নির্দেশটি নানা ধরনের হতে পারে। তবে সবচেয়ে সহজ নির্দেশে কেন্দ্রবিন্দু এবং ব্যাসার্ধের উল্লেখ থাকে।

উদাহরণ 21.

```
100 CIRCLE (100, 80), 30
```

এই নির্দেশ একটি বৃত্ত আঁকবে যার কেন্দ্রবিন্দুর স্থানাঙ্ক হবে (100, 80) এবং ব্যাসার্ধ 30। এখানে উল্লেখ করা প্রয়োজন যে, কেন্দ্রবিন্দুর উপর নির্ভর করে ব্যাসার্ধ এমন দিতে হবে যার ফলে বৃত্তটি আঁকার সময় পর্দার বাইরে চলে না যায়।

উদাহরণ 22.

```
100 CIRCLE (40, 70), 50
```

এবারে কিন্তু বৃত্তটি আঁকা সম্ভব হবে না। এখানে ব্যাসার্ধ 50 আছে এবং কেন্দ্রবিন্দু 70 সংখ্যক লাইনের 40 বিন্দু হওয়ায় বামপার্শ্বে 50 যাওয়া সম্ভব নয়। কাজেই আঁকার সময় ভুল হয়েছে বলে কমপিউটার থেমে যাবে।

সরলরেখার মতই বৃত্তের সীমানাও রং দিয়ে করা সম্ভব।

উদাহরণ 23.

```
100 CIRCLE (100, 80), 30, 2
```

এক্ষেত্রে বৃত্তটির কেন্দ্রবিন্দু (100, 80) স্থানাঙ্ক এবং ব্যাসার্ধ 30। এই বৃত্তের সীমানার রং নির্ভর করবে বর্তমান প্যালিটের উপরে। বর্তমানে প্যালিট 0-থাকলে রং হবে লাল। কিন্তু প্যালিট 1 থাকলে রং বেগুনী-লাল হবে। পটভূমির রং অবশ্যই এর আগে যে COLOR নির্দেশ থাকবে তার উপর নির্ভর করবে।

নীচের উদাহরণের সাহায্যে 5টি বৃত্ত আঁকা হচ্ছে।

উদাহরণ 24.

```
10 SCREEN 1
20 CLS
30 KEY OFF
40 COLOR 0, 1
50 CIRCLE (150, 100), 30, 2
60 CIRCLE (210, 100), 30, 2
```

```
70  CIRCLE (270, 100), 30, 2
80  CIRCLE (185, 125), 30, 2
90  CIRCLE (245, 125), 30, 2
100  IF INKEY$ =" " THEN 100
110  SCREEN 0
120  WIDTH 80
130  END
```

এখানে কালো পটভূমির উপর বেগুনি-লাল সীমানার 5 টি বৃত্ত পাওয়া যাবে। এই প্রোগ্রাম চালানো হলে যে ছবি পর্দায় ফুটে উঠবে তা প্রোগ্রামটি চালিয়েই দেখা সম্ভব।

একটি বাক্সের মধ্যে একটি বৃত্ত আঁকার জন্য নীচের নির্দেশগুলির সাহায্য নেওয়া যেতে পারে।

উদাহরণ 25.

```
10  SCREEN 1
20  CLS
30  KEY OFF
40  COLOR 0, 1
50  LINE (120, 75) –STEP (60, 50), 3, B
60 CIRCLE (150, 100), 25, 2
65  IF INKEY$ =" " THEN 65
70  SCREEN 0
80  WIDTH 80
90  END
```

এবারে কালো পটভূমির উপর একটি আয়তক্ষেত্র থাকবে যার দুই কৌণিক বিন্দুর স্থানাঙ্ক হবে (120, 75) এবং (180, 125) এবং আয়তক্ষেত্রর মধ্যে একটি (150, 100) কেন্দ্রবিন্দু ও 25 ব্যাসার্ধের একটি বৃত্ত পাওয়া যাবে। আয়তক্ষেত্রর চারদিকের রং সাদা এবং বৃত্তের সীমানার রং বেগুনী-লাল হবে।

অনেক সময়ে পুরো বৃত্ত না এঁকে বৃত্তের একটি অংশ আঁকার প্রয়োজন হলে CIRCLE নির্দেশে তারও ব্যবস্থা রাখা হয়েছে।

উদাহরণ 26.

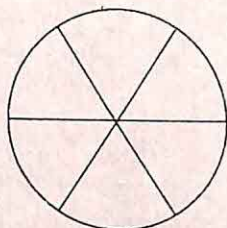
```
100  PI = 3.141593
110  CIRCLE (180, 100), 40, 2, 0, PI/2
```

110 সংখ্যক লাইনের নির্দেশ অনুসারে কমপিউটার (180, 100) কেন্দ্রবিন্দু এবং 40 ব্যাসার্ধের একটি বৃত্তাংশ আঁকবে।

বৃত্তটির যে অংশ আঁকা হবে তা হল 0 থেকে শুরু করে PI/2 পর্যন্ত অর্থাৎ 0 থেকে PI/180 × 90 বা 90° পর্যন্ত বৃত্তের পরিধি আঁকা হবে। এখানে PI-কে π ধরার জন্য PI = 3.141593 নেওয়া হয়েছে। বৃত্তের বাকি অংশ পাওয়ার প্রয়োজন হলে নীচের নির্দেশ দিতে হবে।

উদাহরণ 27.

```
200 CIRCLE (180, 100), 40, 2, PI/2, 0
```

একটি বৃত্তকে 6 ভাগে ভাগ করতে হলে কি ভাবে তা করা যাবে? আমরা জানি যে, একটি বৃত্ত 360°। একে 6 ভাগ করলে প্রতি ভাগে হবে 60°। কাজেই প্রথমভাগ 0° থেকে 60°, দ্বিতীয়ভাগ 60° থেকে 120° এবং এইভাবে ষষ্ঠভাগ হবে 300° থেকে 360°। ছবিটি দেখতে কিরকম হবে তা নীচে দেখানো হচ্ছে।

নীচে বেসিকে নির্দেশের সাহায্যে এই ছবিটি আঁকা হবে।

উদাহরণ 28.

```
10 SCREEN 1
20 CLS
30 KEY OFF
40 COLOR 0, 1
50 PI = 3.141593
60 S = 0
70 FOR I = 60 TO 360 STEP 60
80 A = PI/180 *I
90 CIRCLE (180, 100), 40, 3, S, –A
100 S = –A
110 NEXT I
120 IF INKEY$ =" " THEN 120
130 SCREEN 0
140 WIDTH 80
```

150 END

ভিডিইউ-এর পর্দা দৈর্ঘ্য এবং প্রস্থে সমান নয়। পাশাপাশি 6টি বিন্দুর যে জায়গা লাগে তা উপর-নীচ 5টি বিন্দুর সমান। এর জন্যে বৃত্ত আঁকার সময় বেসিকের কমপাইলার বা ইন্টারপ্রিটার এই প্রতিবিম্বের দৈর্ঘ্য ও প্রস্থের অনুপাত 5/6 ধরে নেয়। এই অনুপাত ধরে না নিলে বৃত্তটি ঠিক মত দেখাবে না। CIRCLE নির্দেশে এই অনুপাত 5/6 হলে তা লেখার দরকার হয় না। কিন্তু উপবৃত্ত আঁকার সময়ে অনুপাত 5/6 এর চেয়ে ছোট বা বড় হবে এবং সেক্ষেত্রে তা CIRCLE নির্দেশে জানানোর প্রয়োজন। নীচের নির্দেশ থেকে এই অনুপাত কিভাবে দিতে হয় তা বোঝা যাবে।

উদাহরণ 29.

100 CIRCLE (160, 100), 40, 2, 0, PI/4, A

এই নির্দেশে A-এর মান 5/6 এর চেয়ে ছোট কিংবা বড় হলে নির্দেশটির সাহায্যে একটি উপবৃত্ত পাওয়া যাবে।

CIRCLE নির্দেশ দেওয়ার নিয়ম হলঃ

CIRCLE (x1, y1), ব্যাসার্ধ, রং-এর সংখ্যা, শুরু, শেষ, অনুপাতের মান

পুরো বৃত্ত বা উপবৃত্ত আঁকার প্রয়োজন না হলে শুরু এবং শেষ দিতে হবে। অন্যথায় না দিলেও হয়। সেক্ষেত্রে এই নির্দেশটি হবে

CIRCLE (x1, y1), ব্যাসার্ধ, রং-এর সংখ্যা,,, অনুপাতের মান

শুরু এবং শেষ উল্লেখ না করলেও কমা চিহ্ন দিয়ে সেই জায়গা ফাঁকা রাখতে হবে এবং এরপর অনুপাতের মান দেওয়া সম্ভব। কেবলমাত্র বৃত্ত আঁকতে হলে অনুপাতের মানও কিছু উল্লেখ করার প্রয়োজন নেই এবং সেক্ষেত্রে এই নির্দেশটি হবে

CIRCLE (x1, y1), ব্যাসার্ধ, রং-এর সংখ্যা

CIRCLE নির্দেশে কেবলমাত্র ব্যাসার্ধের উল্লেখ করতেই হবে। অন্যসব নিয়ামকগুলি প্রয়োজনে করা হয়। এসবের প্রয়োজন না হলে কিছু না লিখলেও ক্ষতি নেই। কিন্তু পরের কোনো একটির দরকারে আগের সব কিছুই উল্লেখ করতে হবে কিংবা কমা চিহ্নের সাহায্যে সেই জায়গা ফাঁকা রাখা যাবে। আংশিক বৃত্তের প্রয়োজনে নির্দেশটি নীচের মত হবে–

CIRCLE (x1, y1) ব্যাসার্ধ, , শুরু, শেষ

এখানে রং-এর সংখ্যা জানানোর প্রয়োজন না থাকলেও কমা চিহ্নের সাহায্যে সেই জায়গা বোঝানো হচ্ছে। নীচের নির্দেশগুলি উপবৃত্তের ছবি পর্দায় আঁকবে।

উদাহরণ 30.

```
100 CIRCLE (160, 100), 40, 2, , , 1/2
```

উদাহরণ 31.

```
100 CIRCLE (160, 100), 40, 2, , , 2
```

PAINT নির্দেশঃ

এই নির্দেশের সাহায্যে একটি অঞ্চল রং করা সম্ভব। এই নির্দেশ দেওয়ার সময় দুটি জিনিস জানার প্রয়োজন।

1. যে অঞ্চল রং করতে হবে সেই অঞ্চলের যে কোনো একটি স্থানাঙ্ক এবং এই অঞ্চলের সীমানার রং।

2. এই অঞ্চলটি কি দিয়ে রং করতে হবে।

উদাহরণ 32.

```
10 SCREEN 1
20 CLS
30 KEY OFF
40 COLOR 0, 1
50 CIRCLE (160, 100), 40, 3
60 PAINT (150, 90), 1, 3
70 SCREEN 0
80 WIDTH 80
90 END
```

এই উদাহরণে প্যালিট 1 থাকায় বৃত্তের সীমানার রং হবে সাদা এবং 50 সংখ্যক লাইনের নির্দেশ অনুসারে এই বৃত্তের কেন্দ্রের স্থানাঙ্ক হবে (160, 100) এবং ব্যাসার্ধ 40। 60 সংখ্যক লাইনের নির্দেশ অনুসারে (150, 90) বিন্দুটি বৃত্তের ভিতরের এবং রং-এর সংখ্যা 1 হওয়ায় বৃত্তের ভিতরের সব জায়গা মাঝারি নীল রং-এ হবে।

PAINT নির্দেশে দ্বিতীয় সংখ্যাটি দিয়ে বোঝানো হয় যে, যতক্ষণ পর্যন্ত এই রংটি পাওয়া না যাচ্ছে ততক্ষণ প্রথম সংখ্যাটি যে রং বোঝাচ্ছে সব জায়গা সেই রং-এর হবে। ভিডিইউ-এর পর্দাকে একটি সরলরেখা দিয়ে দুভাগে ভাগ করা যায়। এই দুভাগে আবার দু ধরনের রং ব্যবহার করা সম্ভব। নীচের উদাহরণের সাহায্যে তা

দেখানো হচ্ছে–

উদাহরণ 33.

```
100 LINE (0, 50) – 319, 50), 3
110 PAINT (10, 40), 1, 3
120 PAINT (10, 80), 2, 3
```

100 সংখ্যক লাইনের নির্দেশ অনুসারে পর্দাটিকে উপর-নীচে দুভাগ করা হচ্ছে। প্রথম ভাগ হবে 0 সংখ্যক লাইন থেকে 50 সংখ্যক লাইন পর্যন্ত এবং দ্বিতীয় ভাগে 50 সংখ্যক লাইন থেকে 199 সংখ্যক লাইন পর্যন্ত থাকবে। এর পরের PAINT নির্দেশে যে বিন্দুটি নেওয়া হয়েছে তার স্থানাঙ্ক (10, 40) হওয়াতে তা প্রথম ভাগে পড়বে। এইভাগে একভাবে এই নির্দেশের সাহায্যে রং করা হল। এবারে 120 সংখ্যক লাইনের PAINT নির্দেশের বিন্দুর স্থানাঙ্ক দ্বিতীয় ভাগে থাকায় এই অঞ্চল আবার অন্য রকম রং দিয়ে রং করা হচ্ছে।

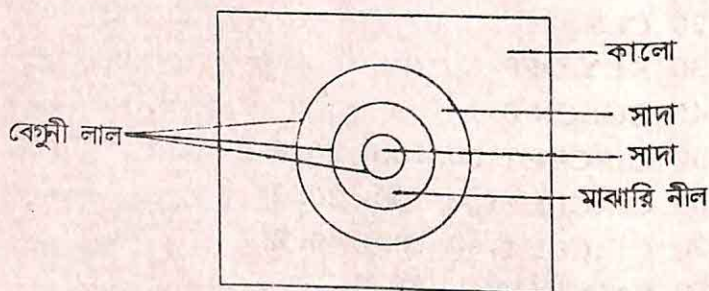
এবারের উদাহরণের সাহায্যে দেখানো হবে কিভাবে তিনটি সমকেন্দ্রিক বৃত্তের বিভিন্ন অঞ্চল নানারকম রং দিয়ে রং করা যায়।

উদাহরণ 34.

```
10 SCREEN 1
20 CLS
30 KEY OFF
40 COLOR 0, 1
50 CIRCLE (160, 100), 20, 2
60 CIRCLE (160, 100), 30, 2
70 CIRCLE (160, 100), 40, 2
80 PAINT (150, 100), 3, 2
90 PAINT (135, 100), 1, 2
100 PAINT (125, 100), 3, 2
110 IF INKEY$ =" " THEN 110
120 SCREEN 0
130 WIDTH 80
140 END
```

এখানে তিনটি সমকেন্দ্রিক বৃত্ত আঁকা হচ্ছে। এই বৃত্তগুলি কেন্দ্রবিন্দুর স্থানাঙ্ক (160, 100) এবং ব্যাসার্ধ যথাক্রমে 20, 30 এবং 40। প্রত্যেকটি CIRCLE নির্দেশে রংয়ের সংখ্যা 2 এবং প্যালিট

1-এর জন্য বৃত্তগুলির সীমানার রং হবে বেগুনী-লাল। এবারে প্রথম PAINT নির্দেশ পালন করার অর্থ ভিতরের বৃত্তের রং সাদা করা কিন্তু কেবলমাত্র ভিতরের বৃত্তের রংই কেন সাদা হবে ? PAINT নির্দেশে প্রথম সংখ্যা 3 সাদা রং নির্দেশ করছে এবং এখানে যে বিন্দুর স্থানাঙ্ক দেওয়া আছে তা ভিতরের বৃত্তের একটি বিন্দু। আর যে মুহূর্তে সীমানার রং বেগুনী-লাল পাওয়া যাবে তখনই ভিতরের বৃত্তের রং করা বন্ধ হবে। দ্বিতীয় PAINT নির্দেশে যে বিন্দুর স্থানাঙ্ক দেওয়া আছে তা প্রথম এবং দ্বিতীয় বৃত্তের মধ্যেকার একটি বিন্দু। এই জায়গাটি অর্থাৎ প্রথম বৃত্ত এবং দ্বিতীয় বৃত্তের মধ্যের জায়গার দুটিকের সীমানায় বেগুনী লাল রং থাকায় কেবলমাত্র এই ভিতরকার জায়গার রং হবে মাঝারি নীল। এরপরের PAINT নির্দেশ অনুসারে দ্বিতীয় বৃত্ত এবং তৃতীয় বৃত্তের মধ্যবর্তী জায়গার রং আবার সাদা হবে।

COLOR নির্দেশে 0 থাকায় পটভূমির রং হবে কালো। এই কালো পটভূমির উপর তিনটি সমকেন্দ্রিক বৃত্ত পাওয়া যাবে। এই বৃত্তগুলির সীমানার রং হবে বেগুনী-লাল। প্রথম বৃত্তটির ভিতরের রং হবে সাদা। এরপর প্রথম এবং দ্বিতীয় বৃত্তের মধ্যেকার জায়গার রং হবে মাঝারি নীল। আবার দ্বিতীয় এবং তৃতীয় বৃত্তের মধ্যেকার জায়গার রং হবে সাদা। ছবিটি দেখতে হবে নীচের মত।

GET এবং PUT নির্দেশঃ

GET নির্দেশের সাহায্যে পর্দার ছবি স্মৃতিতে একটি বিন্যাস হিসেবে সঞ্চয় করে রাখা সম্ভব এবং PUT নির্দেশ ব্যবহার করে ওই বিন্যাসে সঞ্চিত তথ্য পর্দার বিভিন্ন স্থানে প্রকাশ করা যায়।

GET নির্দেশ দেওয়ার নিয়ম হলঃ

GET (x1, y1) – (x2, y2), A

(x1, y1) এবং (x2, y2), পর্দায় দুটি বিন্দুর স্থানাঙ্ক এবং A একটি বিন্যাস চলরাশির নাম। A বিন্যাসের আকার বের করার উপায় নীচে দেখানো হবে।

উদাহরণ 35.

```
100 GET (160, 100) – (190, 120), A
```

আয়তক্ষেত্রটির একদিকের বিন্দুর স্থানাঙ্ক (160, 100) এবং ঠিক তার বিপরীত দিকের বিন্দুর স্থানাঙ্ক (190, 120) হলে আয়তক্ষেত্রটি মোট 31 × 21 বিন্দু দিয়ে তৈরি। কি করে 31 এবং 21 পাওয়া গেল ? 160 থেকে শুরু করে 190 পর্যন্ত গণনায় 31 হয়। ঠিক সেই একইভাবে 100 থেকে 120 পর্যন্ত 21 হবে। এত সংখ্যক বিন্দু রাখার জন্য কত বড় বিন্যাস নিতে হবে ? বিন্যাসের আকার N কত বড় হবে নীচের সূত্র ব্যবহার করে তা নির্ণয় করা সম্ভব।

N = 4 + y INT ((2x + 7)/8)

উপরের উদাহরণে x = 31 এবং y = 21। সুতরাং

N = 4 + 21 × INT ((2 × 31 + 7)/8)
  = 4 + 21 × INT (69/8)
  = 4 + 21 × 8
  = 4 + 168
  = 172 বাইট

এই উদাহরণে বিন্যাস চলরাশির নাম A ব্যবহার করায় বেসিক ভাষায় এই নামের চলরাশিতে 4টি বাইট রাখা যায়। আবার A% ব্যবহার করলে 2টি বাইট এবং A# নামে 8টি বাইট রাখা সম্ভব। সুতরাং A নামের বিন্যাসের আকার হবে 172/4 = 43 অর্থাৎ A(42) লিখলেই হবে। এর ফলে A (0) থেকে আরম্ভ করে A(42) পর্যন্ত মোট 43টি জায়গা পাওয়া যাবে।

স্মৃতিতে সঞ্চিত এই ছবিটি এবারে প্রয়োজনে পর্দায় বিভিন্ন জায়গাতে PUT নির্দেশের সাহায্যে দেখানো সম্ভব। GET এবং PUT নির্দেশের ব্যবহার এবারে একটি উদাহরণের সাহায্যে দেখানো হচ্ছে –

উদাহরণ 36.

```
10 SCREEN 1
20 CLS
30 KEY OFF
40 COLOR 7, 0
50 LINE (160, 100) – (190, 120), 2, BF
60 DIM A (42)
70 GET (160, 100) – 190, 120), A
```

```
80  CLS
90  PUT (40, 10), A
100  PUT (120, 10),  A
110  PUT (200, 10), A
120  PUT (280, 10), A
130  IF INKEY$ =" " THEN 130
140  SCREEN 0
150  WIDTH 80
160  END
```

50 সংখ্যক লাইনের নির্দেশ অনুসারে রং করা এমন একটি আয়তক্ষেত্র পাওয়া যাবে যার বিপরীত কোণের বিন্দু দুটির স্থানাঙ্ক (160, 100) এবং (190, 120)। এই আয়তক্ষেত্রটি এরপর 70 সংখ্যক লাইনের নির্দেশ অনুসারে A নামের বিন্যাসে সঞ্চয় করে রাখা হচ্ছে। এরপর পর্দাটি পরিষ্কার করা হল। 90 সংখ্যক লাইন থেকে আরম্ভ করে 120 সংখ্যক লাইনে 4 টি PUT নির্দেশ দেওয়া আছে। এই PUT নির্দেশগুলির সাহায্যে পর্দার বিভিন্ন জায়গাতে 4 টি ওই একই আকারের আয়তক্ষেত্রর ছবি ফুটে উঠবে। PUT নির্দেশেই বলা থাকছে আয়তক্ষেত্রটির প্রথম বিন্দুটি কোন স্থানাঙ্ক থেকে শুরু হবে। কাজেই PUT নির্দেশে প্রথম বিন্দুটির স্থানাঙ্ক দেবার সময় খেয়াল রাখতে হবে যেন পুরো ছবিটিই পর্দায় ফুটিয়ে তোলা যায়।

PUT নির্দেশের সাহায্যে পর্দায় ছবি ফুটিয়ে তোলার সময় বিন্যাস A-এর রং কি হবে তা নির্ভর করে A এবং পর্দার রং-এর উপর। এই নির্দেশের সাহায্যে একটি বিন্দুর রং কিভাবে করা হয় তা নীচের টেবিলের সাহায্যে বোঝা সম্ভব।

		বিন্যাসের রং			
		0	1	2	3
বর্তমান	0	0	1	2	3
পর্দার	1	1	0	3	2
রং	2	2	3	0	1
	3	3	2	1	0

টেবিল থেকে বোঝা যাচ্ছে যে, একই PUT নির্দেশ পরপর দুবার ব্যবহার করলে পর্দার বিন্দুর যা রং ছিল তাই ফিরে পাবে। কি ভাবে তা বোঝা গেল ? মনে করা যাক বিন্যাসের রং-য়ের সংখ্যা 2 এবং বর্তমান পর্দার রং-এর সংখ্যা 3। সেক্ষেত্রে টেবিল

থেকে পর্দায় এই নতুন বিন্দুর রং PUT নির্দেশ ব্যবহার করলে হবে 1। দ্বিতীয়বার PUT নির্দেশ ব্যবহার করলে এবারে পর্দার বিন্দুর রং 1 এবং বিন্যাসের ক্ষেত্রে 2-ই থাকবে। কাজেই পর্দার নতুন বিন্দুর রং হবে 3 এবং এই 3 প্রথম PUT নির্দেশ ব্যবহারের আগে পর্দার বিন্দুর রং ছিল । অর্থাৎ পরপর দুবার একই PUT নির্দেশ ব্যবহার করে যে ছবিটি প্রথমবারের PUT নির্দেশের পর পাওয়া গিয়েছিল তা মুছে যাবে।

GET নির্দেশের সাহায্যে ছবি সঞ্চয় করার সময়ে ছবির কোনো একটি বিন্দুর যে রং থাকে PUT নির্দেশ ব্যবহার করে সেই রং-এর জায়গায় অন্য কোনো রং আনার দরকার হলে বিভিন্ন উপায় অবলম্বন করা সম্ভব। এরজন্য এই নির্দেশ লেখার নিয়ম নীচে দেওয়া হলঃ

PUT (X, Y), A, action

X, Y এবং A-এর সম্বন্ধে আগেই আলোচনা করা হয়েছে। এবারে action নিয়ে বলা যাক। action পাঁচ রকমের হতে পারে— XOR, PSET, PRESET, AND এবং OR। উপরের টেবিলের সাহায্যে যে ভাবে বিন্দুর রং বের করা হয় তা এবং XOR পদ্ধতি একই। অর্থাৎ নীচের নির্দেশ দুটি একই কাজ করবে।

PUT (X, Y), A এবং PUT (X, Y), A, XOR

XOR ছাড়া অন্যান্যতে PUT নির্দেশ ব্যবহার করে বিন্যাসের বিন্দুর রং যা হবে তা নীচের টেবিলগুলি থেকে বোঝা যাবে।

PSET

		বিন্যাসের রং			
		0	1	2	3
বর্তমান	0	0	1	2	3
পর্দার	1	0	1	2	3
রং	2	0	1	2	3
	3	0	1	2	3

PRESET

		বিন্যাসের রং			
		0	1	2	3
বর্তমান	0	3	2	1	0
পর্দার	1	3	2	1	0
রং	2	3	2	1	0
	3	3	2	1	0

AND

		বিন্যাসের রং			
		0	1	2	3
বর্তমান	0	0	0	0	0
পর্দার	1	0	1	0	1
রং	2	0	0	2	2
	3	0	1	2	3

OR

		বিন্যাসের রং			
		0	1	2	3
বর্তমান	0	0	1	2	3
পর্দার	1	1	1	3	3
রং	2	2	3	2	3
	3	3	3	3	3

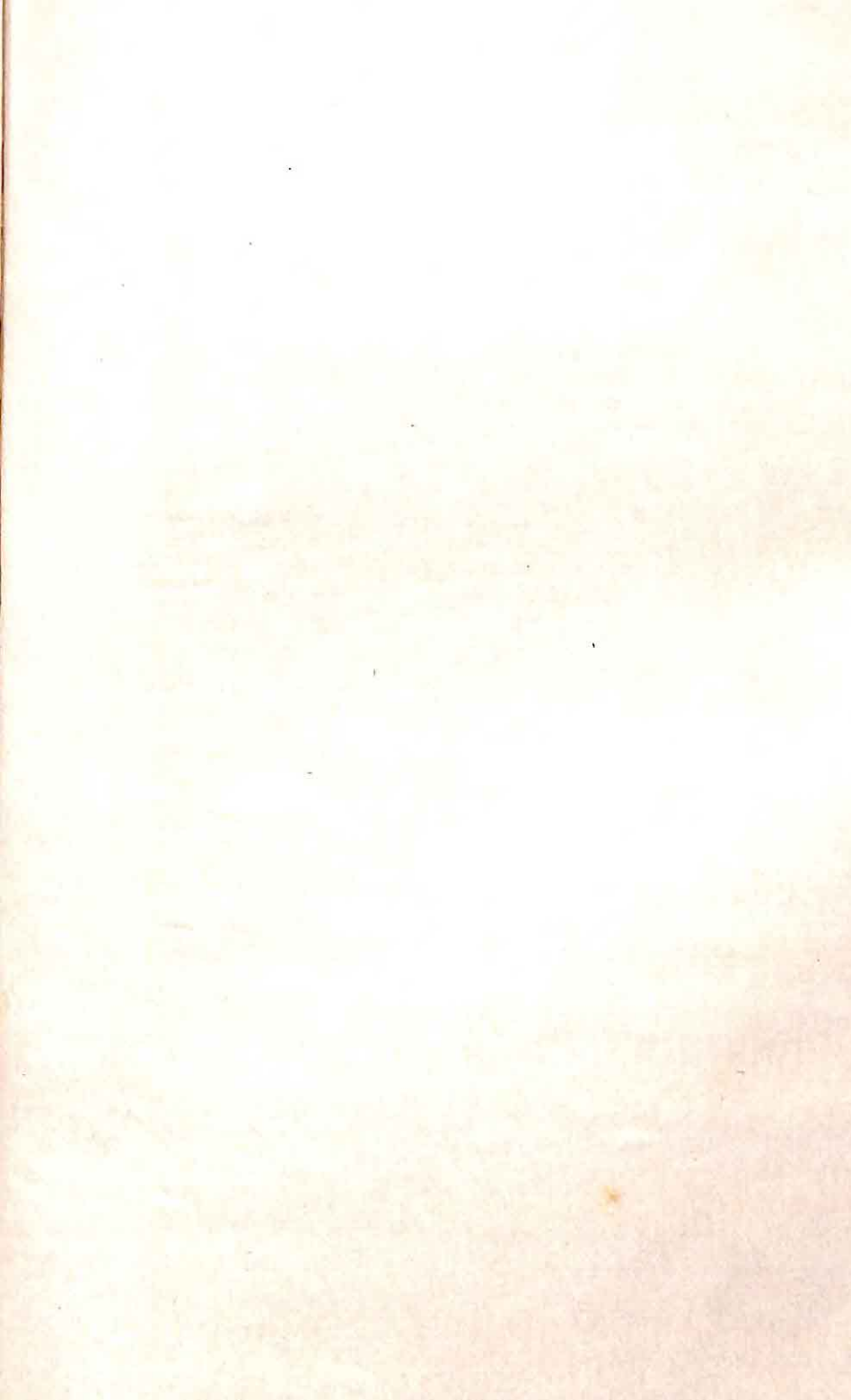
PSET-এর ক্ষেত্রে বর্তমান পর্দার যেকোনো রংই থাক না কেন বিন্যাসের বিন্দুর রং যা আছে তাই নেওয়া হবে। PRESET-এর বেলাতেও PSET-এর মতই বিন্যাসের বিন্দুর রং-এর উপরই নির্ভর করবে। তবে এক্ষেত্রে 3 থাকলে 0, 0 থাকলে 3, 2 থাকলে 1 এবং 1-এ 2 হবে। অর্থাৎ ছবিটির মৌলিক রং যা ছিল তার বিপরীত রং পাওয়া যাবে। AND এবং OR-এর ক্ষেত্রে টেবিল থেকে রং বের করা সম্ভব। এবারে নীচের উদাহরণের সাহায্যে পর্দায় এপাশ থেকে ওপাশে এবং উপর থেকে নীচে একটি বল লাফানোর ছবি ফুটিয়ে তোলা দেখানো হচ্ছে। ছবিটি দেখতে হলে কমপিউটারে প্রোগ্রামটি চালাতে হবে।

উদাহরণ 37.

```
10 SCREEN 1
20 CLS
30 KEY OFF
40 COLOR 0, 1
50 CIRCLE (4, 98), 4, 2
60 PAINT (4, 98), 3, 2
70 DIM A% (16)
80 GET (0, 94) – (8, 102), A%
```

```
 85 FOR J = 1 TO 3
 90 X = 9
100 Y = 94
110 WHILE X < = 311
120 PUT (X, Y), A%
130 FOR B = 1 TO 15
140 NEXT B
150 PUT (X, Y), A%
160 X= X + 4
170 WEND
180 X = 158
190 Y = 4
200 WHILE Y < = 190
210 PUT (X, Y), A%
220 FOR B = 1 TO 15
230 NEXT B
240 PUT (X, Y), A%
250 Y = Y + 4
260 WEND
265 NEXT J
270 END
```

এই প্রোগ্রামটি কি ভাবে কাজ করবে ? 40 সংখ্যক লাইনে COLOR নির্দেশে 0 এবং 1 থাকায় পটভূমির রং কালো এবং প্যালিটের সংখ্যা 1 হবে। এরপর 50 সংখ্যক লাইনের নির্দেশ অনুযায়ী 4 ব্যাসার্ধের একটি বৃত্ত আঁকবে যার পরিধির রং হবে মাঝারি নীল। 60 সংখ্যক লাইনের PAINT নির্দেশের কাজ হল এই বৃত্তটির ভিতর সাদা রং করা। এরপর এই বৃত্তটি GET নির্দেশের সাহায্যে A% নামের বিন্যাস চলরাশিতে সঞ্চয় করা হল। 120 সংখ্যক লাইনে PUT নির্দেশের সাহায্যে এই বৃত্তটি পর্দায় 94 সংখ্যক লাইনে দেখানো হচ্ছে। বৃত্তটির কেন্দ্র (9, 94) স্থানাঙ্কে নেওয়া হয়েছে। এরপর 130 এবং 140 সংখ্যক লাইনের নির্দেশগুলি লক্ষণীয়। এদের এখানে কেন দেওয়া হয়েছে ? এই দুটি নির্দেশ পালন করতে কমপিউটারের এক সেকেণ্ডের থেকেও কম সময় লাগবে। ওই সময় পর্যন্ত বৃত্তটিকে পর্দায় দেখা যাবে। এই দুটি নির্দেশ না থাকলে পর্দায় বৃত্তটি দেখা সম্ভবপর হোত না। বৃত্তটি

আঁকার সঙ্গে সঙ্গেই 150 সংখ্যক লাইনের নির্দেশ অনুযায়ী পরপর দুবার একই PUT নির্দেশ থাকায় বৃত্তটি মুছে যেত। আরও কিছু সময়ের জন্য বৃত্তটিকে ধরে রাখার প্রয়োজনে। 130 সংখ্যক লাইনের নির্দেশে 15-এর জায়গায় আরও কোনো বড় সংখ্যা লিখতে হবে। 150 সংখ্যক লাইনের নির্দেশ পালন করার পর বৃত্তটির কেন্দ্র 94 সংখ্যক লাইনেই 4 ঘর ডানদিকে সরিয়ে নিয়ে আবার 120 সংখ্যক লাইনের নির্দেশ অনুযায়ী এই নূতন বৃত্তটি পর্দায় দেখানো হচ্ছে। এইভাবে ওই একই লাইনে 4 ঘর করে সরিয়ে একটি বৃত্ত পর্দায় ফুটে উঠে মিলিয়ে যাবে। এর ফলে মনে হবে একটি বল পর্দায় বামপাশ থেকে ডানপাশে সরে যাচ্ছে। ওই লাইনে যতক্ষণ পর্যন্ত না X-এর মান 311-এর থেকে বড় হচ্ছে ততক্ষণ পর্যন্ত একটি বৃত্ত 4 ঘর করে সরিয়ে আঁকা হবে। এরপর আবার X-এর মান 158 নিয়ে Y-এর মান 4 থেকে শুরু করে একটি বৃত্ত পর্দায় ফুটে উঠবে। অর্থাৎ এবারে পর্দায় উপর থেকে নীচে একটি বল সরে সরে যাচ্ছে দেখা যাবে। এক্ষেত্রে বৃত্তটির কেন্দ্রর X-এর মান সব সময়েই 158 হবে এবং Y-এর মান 4 করে বেড়ে 190-এর থেকে বড় হলে থামবে। এরপর 265 সংখ্যক লাইনের নির্দেশের জন্য আবার 85 সংখ্যক লাইনে ফিরে যাবে। এখানে FOR নির্দেশে J = 1 TO 3 থাকায় প্রথম থেকে মোট 3 বার বামপাশ থেকে ডানপাশে এবং উপর থেকে নীচে একটি বৃত্ত সরে যাচ্ছে দেখা যাবে।

কমপিউটার পরিচিতি ও বেসিক ভাষা

কমপিউটার কথাটা আজ জড়িয়ে রয়েছে আমাদের সমস্ত জীবনের সঙ্গে । চিকিৎসাবিজ্ঞান, মহাকাশবিজ্ঞান থেকে শুরু করে ভাগ্য নির্ণয় পর্যন্ত বিভিন্ন বিষয় আজ নির্ণীত হচ্ছে কমপিউটারের মাধ্যমে । কিন্তু কমপিউটার নিয়ে কাজ করার প্রাথমিক পাঠ হলো বেসিক ল্যাঙ্গুয়েজ – এই বেসিক ল্যাঙ্গুয়েজই হলো এ বই-এর বিষয়বস্তু । এই বইয়ের মধ্যে দিয়ে কমপিউটার-এর প্রথম শিক্ষার্থীরা যেমন কমপিউটার সম্পর্কে একটা প্রাথমিক ধারণা পাবেন তেমনি কমপিউটারের বেসিকসহ ছোট ছোট প্রোগ্রামও নিজেরা করতে পারবেন । এ ছাড়াও বেশ কিছু প্রোগ্রাম উদাহরণ হিসেবে দেওয়া হয়েছে এই বইতে–যা শিক্ষার্থী পাঠকরা কমপিউটারে চালিয়ে দেখে নিতে পারবেন ও গ্রাফিক্সের ছবি আঁকতে পারবেন ।

লেখক ডঃ দেবব্রত ঘোষদস্তিদার যাদবপুর বিশ্ববিদ্যালয়ের কমপিউটার সায়েন্স এ্যান্ড ইঞ্জিনিয়ারিং বিভাগের অধ্যাপক । তিনি কমপিউটার শিক্ষার্থীদের বহু সমস্যার সমাধান করেছেন, বর্তমান বইটিতে বাংলার মাধ্যমে সেই সব সমস্যার সমাধানের চেষ্টাই করেছেন ।

বিজ্ঞানে রবীন্দ্রপুরষ্কার প্রাপ্ত লেখক অরূপরতন ভট্টাচার্য এই বই -এর যুগ্ম লেখক । ডঃ ঘোষদস্তিদারের কমপিউটার সম্পর্কে আলোচনা কে পাঠকের কাছে আরও হৃদয়গ্রাহী ও প্রাঞ্জল করে তুলতে সাহায্য করেছেন তিনি । বলা যায় বর্তমান বইটি এই দুই লেখকের এক অনন্য যুগলবন্দী ।

প্রচ্ছদ ঃ সৌমেন মুখার্জ্জী

মূল্য – ৪৫ টাকা

বেস্টবুকস্

১এ, কলেজ রো, কলিকাতা - ৭০০ ০০১ ISBN 81-85252-28-9